AFIT/GOA/ENS/99M-02

ANALYSIS OF A NON-TRIVIAL QUEUEING NETWORK

THESIS

Kelly Scott Bellamy, Captain, USAF

AFIT/GOA/ENS/99M-02

Approved for Public Release; Distribution Unlimited

DTIC QUALITY INSPECTED 2

19990409 031

Disclaimer

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Thesis Approval

STUDENT: Kelly Scott Bellamy, Captain, USAF **CLASS**: GOA-99M

TITLE: ANALYSIS OF A NON-TRIVIAL QUEUEING NETWORK

DATE: 01 March 1999

COMMITTEE:	**NAME/TITLE/DEPARTMENT**	**SIGNATURE**
Co-Advisor	W. Paul Murdock, Major, USAF Assistant Professor Department of Operational Sciences	W. Paul Murdock, J
Co-Advisor	Thomas Reid, Major, USAF Deputy Head Department of Mathematics and Statistics	Thomas F. Reid

AFIT/GOA/ENS/99M-02

ANALYSIS OF A NON-TRIVIAL QUEUEING NETWORK

THESIS

Presented to the Faculty of the Graduated School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Kelly Scott Bellamy, BS

Captain, USAF

February 1999

Acknowledgments

To my wife, Judy, for enduring another adventure.

I extend special thanks to all the members of the library staff. Without their support, this work would be sorely lacking. Two former members deserving special recognition are Kathy and Margaret—their professionalism, proficiency, and expediency was truly unbelievable. Additionally, two current members having equal caliber, Donna and Chris, made the task enjoyable by sheer will of their personalities.

I would also like to thank my committee. Their guidance and assistance was second to none. However, it was their reminder that I could not solve the world in six months that proved most beneficial.

Table of Contents

Acknowledgments ii

Table of Contents iii
List of Figures and Tables v

Abstract vi

Executive Summary vii
Overview vii
Research Overview and Findings viii
Limitations xv
Recommendations xvi

I. Introduction 1
Background 1
Scope 3
Overview of Methodology 5
Organization of this Report 5

II. Literature Review 6
Exact Analytic Solutions 6
Product-Form Theorem 14
Approximation Techniques 15
Multiple Customer Class Considerations 27

III. Development and Findings 30
Review of Objective 30
Summary of Scope 31
Overview of Methodology 32
Highlights of Literature Review 32
Selection of Approximation Algorithm 33
A Trial-Size Network (Base Case) 34
Simulation Model for the Base Case 37
Comparison of Estimates for the Base Case 41
Including Customer Classes, I 45
Including Customer Classes, II 49
Including Customer Classes, III 52
Including Customer Classes, IV 54
Going Over the Main Points 56
Using the Procedure to Focus a Simulation Study 58
Methodology Flowchart 69

IV. Conclusion 71
Summary 71
Research Overview and Findings 72
Limitations 75
Recommendations 77

Appendix A: Exact Solution for Base Case 78

Appendix B: Two-Node Network Simulation Model 81

Appendix C: Two-Node Base Case Simulation Output 87
Case 1: Base Case Simulation (10,000 Departures with Replication) 87
Case 2: Base Case Simulation (200,000 Departures, One Run) 89

Appendix D: Reconstruction of Queueing Network Analyzer (QNA) 91

Appendix E: Two-Node Class-Dependent Simulation Output 97
Case 1: Different Service Rates at Queue 1 (200,000 Departures) 97
Case 1a: Different Service Rates at Queue 1 (10,000 Departures, 10 Replications) 99
Case 2: Different Service Rates at Both Queues (50,000 Departures, 10 Replications) 101

Appendix F: Evaluation Network Simulation Model 103
Basic Model 103
Modifications 106

Appendix G: Evaluation Network Simulation Output 107
Case 1: 200,000 Departures 107
Case 2: 250,000 Departures with 10 Replications 109

Bibliography 112

VITA 116

List of Figures and Tables

Figure 1. Two-Node Network [46: 412] 35
Figure 2. Simulation Representation of Nodes 38
Figure 3: Evaluation Network 58
Figure 4: Ways to Study a System 69
Figure 5: Flowchart of Methodology Process 70

Table 1: Base Case Results 42
Table 2: Tightly-Coupled with Heavy Traffic 45
Table 3: Queue 1 Arrival Process 48
Table 4: Waiting Time by Class 51
Table 5: Queue 1 Service Process 54
Table 6: Different Service Rates at Node 1 55
Table 7: Different Service Rates at Both Nodes 56
Table 8: Routing Matrix 60
Table 9: Arrival and Service Information 62
Table 10: QNA Congestion Estimates 62
Table 11: QNA Congestion Estimates Compared to Simulation 63
Table 12: Node 14 Expected Wait 64
Table 13: Node 14 Service Times 65
Table 14: Congestion Estimates under Replication 66
Table 15: Node 14 Expected Wait 67
Table 16: Congestion Estimates under Delayed Feedback 68

Abstract

In studying complex queueing networks, one generally seeks to employ exact analytic solutions to reduce burden on computational resources. Barring the existence of an exact solution, the alternatives include approximation techniques and simulation. Approximation is the more attractive alternative from a time and effort perspective; however, cases exist that are not amiable to this technique.

This work increases the flexibility of approximation techniques in obtaining estimates of congestion measures for complex, open queueing networks having several customer classes and class-dependent structures. We accomplish this by providing the procedure to aggregate multiple classes into a single class in order to apply an existing approximation technique. The resulting method is shown to yield good agreement with results obtained by simulation.

The immediate application of this work is as a tool to focus a simulation study of multi-class queueing networks. For large networks, reasonable performance estimates can be obtained quickly. Once the basic input parameters are determined, different scenarios may be rapidly evaluated in a fraction of the time needed to modify a typical simulation model. This allows one to check ideas and determine where to invest time and funding when constructing a simulation model to obtain performance estimates on a by-class basis.

Executive Summary

Overview

In the study of complex queueing networks, one generally seeks to employ exact analytic solutions to reduce the burden on computational resources. Barring the existence of this solution, the alternatives include approximation techniques and simulation. Approximation is the more attractive alternative from a time and effort perspective; however, cases exist that are not amiable to this technique.

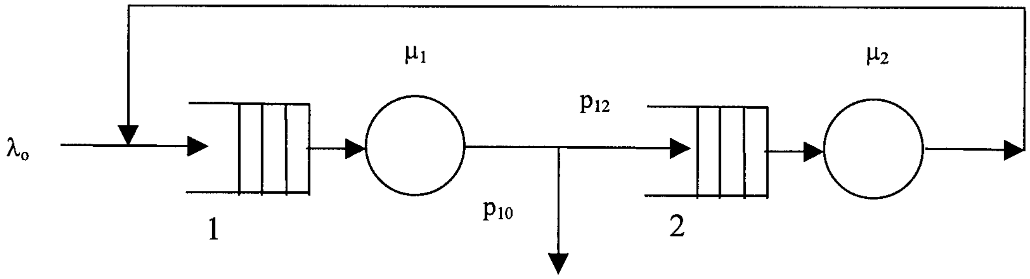
The objective of this work is to increase the flexibility of approximation techniques in obtaining estimates of congestion measures for complex, open queueing networks. The focus of this study is on networks having several customer classes and class-dependent service times. In this case, the restrictions of the network prevent an exact solution and hamper existing approximation schemes. The conditions generally require using simulation to assess network performance. Thus, the goal is to provide a tool to easily obtain these estimates before building an intricate simulation model.

In most simulation studies, you want to build a model having the least complexity necessary to accomplish the analysis tasks. You may be interested in collecting detailed information on different customer classes, but you only want this at nodes meeting some criteria (e.g., longest queue). To do this, you could build a basic model and add class distinctions as you learn more about how the network operates. However, this could become time consuming, particularly in verifying routing paths.

The outcome of this effort is a methodology that allows you to quickly identify candidate locations requiring higher resolution results before building the simulation model. We accomplished this by developing a procedure that aggregates multiple classes into a single class in order to apply an existing approximation technique. We found that the methodology results in good agreement with simulation models that explicitly represent multiple customer classes. The result is a tool that one can use to focus a simulation study. It provides easy evaluation of a particular network configuration, as well as rapid evaluation of alternative configurations.

Research Overview and Findings

The network models utilized in this effort are classic *Jacksonian* networks. These networks contain a finite number of unlimited capacity, single-server queue nodes connected in an arbitrary fashion. Routing between the queues is based on specified branching probabilities. The networks are open: customers enter from the outside world and all customers eventually depart. The service discipline at all queue nodes is first-come-first-serve (FCFS); however, the different customer classes may have different service distributions at any given node. The schematic below illustrates an example of one network structure studied.



The external arrival rate is λ_o, and may represent many classes. The service rates are μ_1 and μ_2 for nodes 1 and 2, respectively, which may depend on class. A customer may depart after node 1 with probability p_{10}. All node 2 customers return to node 1.

A review of relevant literature did not identify an exact analytic solution for the case described. The use of FCFS queues with class-dependent service times prevents obtaining an exact solution. Specifically, networks constructed as given do not have a particular type of solution known as the product-form. However, we also found that product-form and exact solution are synonymous except in rare circumstances. Here, in rare circumstances, small networks may be solved using their global balance equations. For this, the practical limit is on the order of a two-node network, but even that size may become too complicated in some cases.

Satisfied that an exact solution was unattainable, we reviewed various approximation techniques. For networks having service centers that are not continuously busy, decomposition algorithms provided reasonable estimates of network performance and afforded the greatest flexibility for modification. In particular, the algorithm for the Queueing Network Analyzer (QNA, [49]) was found suitable enough to serve as the basic structure for this study. For many networks, QNA can provide congestion measure estimates with errors not exceeding 10 percent relative to simulation values [50].

QNA allows for multiple customer classes with different service time distributions; however, the user has to specify the exact route of each class. For general routing, such as used here, QNA only permits one customer class. Therefore, the task became one of combining general routing and multiple classes.

The author of QNA provided one concept of how to accomplish this combining task, and suggested equations that could be incorporated into the original QNA [51]. However, we found that we could not properly match simulation observations to the proposed equations. We observed the problem to be related to the effect of customers re-circulating in a network (commonly called feedback). From the previous figure, note that a customer may visit node 1 more than once. The QNA algorithm appears to transform a network into a series of serially connected queues, whereas the simulation maintains a representation closer to the actual layout. The result is that the arrival, service, and departure measures found through simulation and calculated by QNA do not have an obvious mapping to each other.

From simulation results used to evaluate the paper mentioned in the previous paragraph, we noted some basic relationships in the data appeared useful. We observed that the probability of a customer class at a given queue followed a simple form. The equation below shows the result for two customer classes having derived arrival rates, $\lambda_{i,cx}$, at some node i for class x. The arrival rates for each class at each node are obtained by solving a system of equations using the external arrival rates and a matrix that specifies routing within the network.

$$P(class\_1 \mid \lambda_{i,c1}, \lambda_{i,c2}) = \frac{\lambda_{i,c1}}{\lambda_{i,c1}+\lambda_{i,c2}}$$

From conditional probability, we then observed that we could form the mean and variance for the aggregate service time of several customer classes having different individual service distributions at a given node. Using the result above and the expected service time given that a customer is of a particular classes, we obtain that the aggregate expected service time at a node is

$$ET_i = p_{i,c1}ET_{i,c1} + p_{i,c2}ET_{i,c2} \quad .$$

In a similar manner, we also find the second moment of the service time, ET_i^2 (replace all ET_i with the squared term in the above expression). Here, you use the analytic expressions for the variance and mean of each class's service distribution in the identity $ET^2 = Var(T) + (ET)^2$ to form each component of the right hand side of the expression. For example, if the service times at node i for each class x are distributed exponentially with rate $\mu_{i,cx}$, then the expected service times and variance are $ET_{i,cx} = 1/\mu_{i,cx}$ and $Var(T) = 1/\mu^2_{i,cx}$, respectively. The terms on the right hand side become $ET^2_{i,cx} = 2/\mu^2_{i,cx}$. This technique is valid for any continuous theoretical distribution, but you only want to use non-negative, or properly truncated, distributions since these are service times. Once we have ET_i and ET_i^2 for the aggregated customer class, we can employ our technique, as discussed next.

We hypothesized that this information could be used in the QNA framework to represent multiple customer classes as a single composite class. One major assumption of QNA is that the mean and squared coefficient of variation (SCV) contain adequate information to represent arrival, service, and departure processes. Where a SCV is the variance divided by the square of the mean. The validity of this assumption was proven prior to the development of QNA. Therefore, the specific structure of QNA seeks to enhance estimates using the assumption. The accuracy of QNA is documented in [50].

Our procedure uses QNA to calculate the arrival and departure SCVs based on the input. Since QNA only views service processes in terms of the mean and SCV at some node, we simply replace single-class expected service times and SCVs with the values obtained by aggregating the class-dependent service processes. Using ET_i and ET_i^2 shown previously, the expression for finding the service SCV, c^2, is obtained as

$$c_{s\,i}^2 = \frac{ET_i^2}{(ET_i)^2} - 1 \quad .$$

To evaluate whether the distribution of our composite customer class in QNA represented reality as viewed by the simulation, we constructed networks having explicit representation of customer classes. At worst, we found that our multi-class QNA was within the performance documented in [50]. At best, we noted that the multi-class QNA differed less than 1 percent from the simulation. This was an evaluation, not a formal test of the hypothesis.

As an example, refer back to the network schematic previously shown. As given in [50], this specific network structure presents the greatest difficulty for the original QNA without multiple classes. We set up a scenario were two customer classes arrive to the network. For one case, each class has exponential service times, but they have different means at each node, as shown below.

Node	Class 1 Rate	Class 2 Rate
1	1	2
2	3	1

The input external arrival rates to node 1 and the derived internal arrival rates at each node are shown next. The internal arrival rates depend on the value for the probability of depart between nodes 1 and 2, which we selected as $p_{10} = 0.30$.

Node	Aggregate	Class 1 Rate	Class 2 Rate
External at 1	1/6	1/10	1/15
1	1/1.8	1/3	1/4.5
2	1/2.57	1/4.29	1/6.43

The actual arrival and service rates observed in simulation will differ slightly. The probability for a customer being of class 1 is calculated at 0.60, which also may differ to a small extent.

The following table shows a comparison between the QNA results using the proposed methodology and a simulation that explicitly represents both customer classes. For notation by node, EN_i is the expected number waiting in a queue, EW_i is the associated expected wait, and ER is the expected response time for the network (average time in the network).

Congestion Measures	Simulation		QNA
	50K Depart, 10 Reps		
	Mean	Std Error	Mean
EN_1	0.387	0.002	0.3935
EW_1	0.697	0.004	0.7084
EN_2	0.087	0.000	0.0935
EW_2	0.225	0.001	0.2404
ER	6.910	0.018	6.9888

Even though this specific network is difficult for QNA, there is good agreement between the estimates. The largest discrepancy with respect to simulation is the expected number waiting at node 2. The error is approximately 7.5 percent if we accept the simulation as representing the *true* state with only 50,000 departures and 10 replications (runs). This is within the documented performance of the original QNA.

We find that the upper limit of the individual 95 percent confidence intervals on the simulation estimates fall below the QNA values. There are two issues here, one of which relates to the length of the simulation runs. QNA is an estimator of long run, or steady state, performance. Therefore, the user must be aware that short-run simulation estimates may differ because of initial conditions. By increasing the length of the simulation runs, we find that the confidence intervals eventually cover the QNA estimates. However, another issue impacts whether we actually reach convergence. That issue relates to the SCV calculations of the feedback stream.

In the instances where we did not get an exact match between the simulation and the multi-class QNA, we found that QNA faithfully represented the performance trend and correct magnitude. For example, say that simulation yields an average of 14.65 customers waiting in a given queue; the multi-class QNA may report 15.03.

Inconsistencies in estimation generally fall into two categories: heavy traffic problems and departure SCV estimation. In any situation where a queue experiences heavy traffic, the simulation and QNA estimates will not likely agree. This is caused more by the initial conditions and the length of the simulation run than with the formulations contained in QNA. However, problems may also occur under moderate traffic if the simulation run is too short to eliminate the startup transient.

We did note that departure SCVs calculated using our service SCVs (a procedure of the QNA algorithm) introduced actual bias into our results. This becomes particularly true when the network has feedback streams. Although the bias is within the bounds noted, this is a problem area requiring modification in a manner suggested by [51]. The previous table contains bias in both columns due to short simulation run length for one and inaccuracies in arrival SCV calculations for QNA.

As a tool to focus a simulation study of a complex network, the method performs well. For large networks, reasonable performance estimates can be obtained quickly. Once the basic input parameters are determined, different scenarios may be rapidly evaluated in a fraction of the time needed to modify a typical simulation model. This allows one to check ideas and determine where to invest time and funding when constructing a simulation model to obtain performance estimates on a by-class basis.

Limitations

The methodology to admit multiple customer classes under probabilistic routing performs well in the QNA framework. Any number of classes may be combined into one composite class before applying the QNA algorithm. Moreover, the results are comparable (sometimes identical) to simulation estimates of congestion measures. However, we have not been able to later decompose the composite customer to give performance measures on a by-class basis. An apparent relationship to the probability of the class of a customer arriving at a queue was noted; however, a working theory did not evolve.

The method to aggregate multiple customer classes may fail if the service rate of one class is significantly higher than other classes. This becomes apparent in calculating the composite service SCV when a negative value is returned. In simple terms, the class having a lower rate dominates the service distribution at that queue—the queue does not seem to exist for the higher rate class. Thus, we develop an apparent inconsistency best resolved by simulation. QNA cannot handle zero-time service centers in general. However, this does not preclude using composite rates that are large. We sought to avoid this situation since it provided no useful insight other than as a limitation.

QNA only estimates long run, or steady state, network performance. Therefore, it is most useful for networks that operate for extended periods or those not having a notable startup transition phase. Additionally, we did not include the estimates of variability provided within the QNA formulation. Given the stated focus, that was deemed unnecessary.

QNA does not permit queueing disciplines other than FCFS. However, due to the abundance of work on product-form solutions, most other disciplines of practical interest have an exact solution. Additionally, QNA is currently restricted to infinite capacity queues.

As part of the scope of this effort, we restricted the study only to Poisson arrivals. Thus, the effectiveness of the class aggregation technique was not verified for general arrival distributions. QNA was developed to permit arrival distributions other than Poisson [1; 48; 49; 50], and is based on work extending back at least to Kelly [25]. Given that arrival SCVs combine in the same manner as service SCVs, the assumption that the procedure works for general arrival and service distributions is not unfounded.

Recommendations

Obtaining greater accuracy for the congestion measure estimates appears to be possible by improving the departure approximations for the queues. This is the subject of [51]; however, simulation observations are not consistent with that methodology. Understanding that methodology and incorporating it within the QNA framework is the next logical step of this work.

Parallel to improving the methodology of including multiple classes is development of the methodology to obtain congestion measure estimates by class. We observed that there appears to be a connection to the probability of a particular class being at a given queue. However, we could not identify the actual relationship. If one could solve this problem, simulation studies to obtain these measures would not be necessary. That is, assuming one is content with approximations of long run performance to guide the development of a simulation study.

Originally, we desired to incorporate capacitated queueing (blocking with loss) simultaneously with class-dependent service time distributions at FCFS queues. However, it quickly became clear that the task of incorporating class dependencies overshadowed that feature. Thus, one direction of future research would be to incorporate this capability. Although not included in the literature review, we looked at several works in the area of capacitated queueing networks. However, most involved other forms of blocking mechanisms (e.g., repetitive service for a blocked customer). We did uncover some literature that a future researcher might begin with [4; 35; 36; 44].

ANALYSIS OF A NON-TRIVIAL QUEUEING NETWORK

I. Introduction

The objective of this thesis is to increase the flexibility of approximation techniques in obtaining estimates of congestion measures for complex, open queueing networks. That is, to easily obtain these estimates, before building an intricate simulation model, on networks having several customer (job) classes and class-dependent structures. This thesis accomplishes this by providing the procedure to aggregate multiple classes into a single class in order to apply an existing approximation technique, and illustrating the comparability of the results obtained by simulation. The immediate application of this work is as a tool to focus a simulation study of a multi-class queueing network. For example, by identifying potential bottlenecks in the presence of classes, the analyst knows where the simulation model requires increased fidelity to obtain class-dependent congestion measures. However, with further expansion of the theoretical framework, we believe that one can extend this approach to obtain class-dependent congestion measures from the approximation.

Background

Queueing networks exist, literally, everywhere with the most commonly cited examples being computer and communications systems. Analytical models, as opposed to simulation, are preferred to obtain measures of system utilization, throughput, response time, and queue lengths. This preference is motivated by the fact that simulation models generate estimates of these measures, which requires application of statistical procedures to obtain a statement of confidence in the results. Analytic solutions, on the other hand, are mathematically exact (accepting that the model assumptions are congruent with the system under study). This, in turn, creates a perception of a lower cost relative to simulation analysis, as well as one of greater validity over simulation.

However, with rare exception, the characteristics that make complicated networks interesting, or challenging, to study also render them analytically intractable. One problem is that the number of system states becomes too unwieldy to manage computationally. Alternately, as will be discussed later, the problem is because these networks tend to violate conditions necessary to an exact analytic solution.

To contend with this problem of intractability, researchers developed approximation techniques to obtain congestion estimates. Without getting into detail here, these techniques are generally classified as decomposition (aggregation) or diffusion methods. The choice of which technique to use depends on the characteristics of the system under study, such as loading (i.e., for systems with continuously busy servers, diffusion approximations provide better estimates [28: 217]).

While approximation techniques provide greater flexibility over sometimes-insoluble analytic solutions, they too have limitations. Most obvious is that they are approximations...sometimes they over-estimate, under-estimate, or fail spectacularly. However, these techniques can provide good estimates when properly used.

Another problem that exists in approximation techniques is the ability to handle multiple classes of customers (jobs) in a general context. Of particular interest to this effort are networks involving probabilistic routing with class-dependent service rates and multiple visits to a service center with overtaking permitted (a latter arrival may pass an earlier arrival). A review of the published works reveals that these networks are problematic to approximation techniques because of the class dependencies, and that this area requires further development.

Actually, no published works found on approximations allow for general routing with class-dependent service rates. Thus, the objective of this work is to increase the flexibility of approximation techniques in providing estimates of congestion measures in a general queueing network under the presence of multiple customer (job) classes and class-dependent service rates.

Scope

The network model utilized in this effort is, in appearance, the classic *Jacksonian* network [20; 21]. Our network contains a finite number of single-server, infinite capacity, queue nodes (waiting line and server combination) connected in an arbitrary fashion. Routing between the queues is based on specified branching probabilities, and customers may visit a node any number of times. The network is open: customers (jobs) enter from the outside world and all customers eventually depart.

While any number of distinct customer classes may enter the network, we use only two for this effort. The external arrival processes for each class are taken separately to be Poisson, but this is not a hard requirement. For our purposes, entry to the network is restricted to a single node; however, this is for convenience since the capability exists to allow entry at all nodes. In general, customers may depart the network after completing service at any node.

The service discipline at all queue nodes is first-come, first-serve (FCFS). However, the different customer classes may have different service distributions at a given node. We use the exponential distribution with differing class-dependent rates. For other service disciplines (i.e., last-come, first-serve; processor sharing), exact analytic solutions exist under this condition. However, for the particular combination of FCFS with different service rates, an exact analytic solution does not exist [40: 72-75]. And, as mentioned previously, none of the approximations permit this condition, either.

Regarding estimates of the congestion measures obtained from approximation, these are for long-run operation of the network. Transient, or startup, behavior is not addressed. Since our objective is to produce a tool to focus a simulation study, this does not pose a problem. However, for networks that cannot realize long-run performance, the estimates have limitations similar to those of exact analytic solutions.

As a final note on the scope of this effort, the reader is asked to remember that this is a groundwork effort in an area for which no published works were found. That being said, this effort is not a demonstration of mathematical prowess. Additionally, it is not a simulation study, nor a sensitivity analysis. Rather, it is the development and demonstration of a practical tool for the analyst.

Overview of Methodology

The below list outlines the general methodology. Specific details are presented later in the chapter on development.

- Determine the state of current research
- Validations of approximation and simulation models
- Extend simulation to more general model
- Demonstrate how approximation can be used to guide a simulation study

Organization of this Report

Chapter 2 contains a review of relevant literature. Chapter 3 presents the development and findings, and ends with a demonstration of how one could use the results of this effort to focus a simulation study. Chapter 4 discusses further research suggestions with the conclusion.

II. Literature Review

The intent of this chapter is to survey works relevant to the objective. While completeness is desirable, we omit sources that focused on closed networks and those that failed to provide additional insight. The first section covers the history of exact analytic solutions, ending with conditions necessary to achieve a solution. Then, we address approximation techniques developed to analyze cases where no exact solution exists or where the solution is computationally intensive. In the final section, we look at some additional issues related to class-dependent structures in a queueing network.

Exact Analytic Solutions

The formal study of queueing networks is quite young, although the development of Queueing Theory began in 1909, when A. K. Erlang published "The Theory of Probabilities and Telephone Conversations." From this, and later works, Erlang spawned the concept of stationary equilibrium, which is an assumption inherent and necessary to most analytic solutions (we will return to this point near the end of this section). In brief, the assumption is that the probability distribution of the system state has a steady state, or limiting, form [17: 52; 39: 546]. While this is not the only assumption needed for an analytic solution, it plays a significant role. Consequently, in these works, Erlang laid the groundwork for obtaining analytical solutions to queueing systems [17: 10-11].

Over the 50 years following Erlang's first paper, research centered mainly on characterizing a single queue node (the waiting line and server) operating in isolation. While the results were necessary to fundamental understanding, no preeminent works existed for networks of queues; although in 1954, J. R. Jackson first considered two queues connected in series [32: 232]. In 1957, however, Jackson presented a paper entitled "Networks of Waiting Lines," which many regard as the seminal work in the study of queueing networks.

In his paper [20: 518], Jackson presented a model for the departments of a machine shop. Here, each department contained different machine types (one or many), and waiting lines formed to gain entrance to each department. Additionally, the workload at each department could come from another department or from outside the shop. Routing between the departments, or out of the shop from a department, was probabilistic.

Keeping with the previous citation, but generalizing the terminology: Jackson studied a network containing a finite number of first-come-first-serve (FCFS) queue nodes, where each has a finite number of servers working at the same rates. As noted, routing between the queues is by specified branching probabilities, and the queue lengths have no limit. However, Jackson stipulated that external arrival be governed by a Poisson process, and that the service times be exponentially distributed. (Aside: It is apparent that our model, as discussed in the first chapter, bears a strong resemblance to the *Jacksonian* construct.)

In analyzing his model, Jackson found that the steady state distribution of the system state is the product of the distributions for each department. That is, the distributions of each department behaved as though they were operating independently from one another [20: 518, 520]. Thus, you can study each department in isolation, and form the joint distribution from the product of each. In addition, it is instructive to note that Jackson appealed to the concept of stationary equilibrium in developing his solution.

This short, four-page presentation by Jackson provides the basic framework needed to obtain analytic solutions. While details of a specific network may vary, analytically obtaining measures of congestion begins by determining the *product-form solution* of the system state distribution under the assumption of stationary equilibrium. “With few exceptions, networks that are analytically tractable are product-form networks…that admit an invariant distribution…[, and t]here are relatively few results known to hold for general non product-form networks…[47: 545, 569].” Though somewhat understated, this observation is vastly significant to this effort and in general.

Extensions to Jackson’s original work are many. In fact, Jackson himself added state-dependent external arrival rates and service rates to maintain a steady flow in the network [21]. However, it was not until 1975 that the second evolution in queueing network theory occurred. Up to this point, the solutions allowed only for a single class of customers, and no method existed to permit distinction. For greater utility, a technique was needed that allowed multiple classes having different arrival rates and different service rates at a given queue node, as well as an allowance for different queueing disciplines.

Baskett, Chandy, Muntz, and Palacious provided such a technique, which also produces the product-form. In fact, Jackson’s result is a special case of the class of problems now referred to as *BCMP* queueing networks (Baskett and co-author initials). “The paper of Baskett et al is, perhaps, the most widely cited paper in the field and may be viewed as the main archival reference for equilibrium results on product-form queueing networks [11: 32]."

What did BCMP give us? This work showed that product-form solutions exist for multi-class *Jacksonian*-like queueing networks. Class-dependent service rates were possible for three of the queueing disciplines presented. Additionally, through the existence of a *class-switching* feature, one could model a variety of routing schemes, as opposed to just probabilistic. Taking these features together, it is possible to model general networks using this method—with some restrictions.

One restriction, for all queuing disciplines, is that the service time distributions must have a rational Laplace transform to attain a solution [3: 251]. With that, the set of queueing disciplines allowing flexibility in class-dependent features were processor sharing (PS), preemptive-resume last-come-first-served (PRLCFS), and infinite server (IS). Though the requirement for the transform sounds strict, the disciplines listed allow general service time distributions.

Flexibility aside, the FCFS queueing discipline (single server is implied here and from now on) did not garner the same generality as the other disciplines. Stated in the abstract, we have: "At first-come-first-served-type service centers, the service time distributions must be identical and exponential for all classes of customers [3: 248]." This begs the question: Was this for convenience, or was it an absolute?

In the case of the FCFS discipline with class-dependent service rates, there is an implication that a solution does not exist unless all classes have exponentially distributed service times with the same mean. Although not given a proof, Basket et al couch the discussion in terms of a property called *balance*, which relates to the rate of flow into and out of a given system state [3: 251-253]. The relevance of this is that the existence of global balance (for the network) is a necessary condition to obtain the product-form solution [32: 238]. In the work, BCMP indicate that it may not be possible to actually obtain the solution to the global balance equations in this particular situation. That is because the independent balance equations (pertaining to the mutually exclusive possible state changes of a queue) are inconsistent. Therefore, even when this case has global balance, we may not be able to solve the equations or produce a product-form solution.

Although BCMP does not explicitly show the lack of a solution to the global balance equations, one reaches the supposition that no solution exists for FCFS queues unless all classes have equal exponential service times. Indeed, many authors lead you to this same belief in their review of BCMP (e.g., Noetzel [34: 791]). However, it must be clear that BCMP did not explicitly make this statement as a finding in the body of the paper. Rather, the restriction on FCFS service distributions is given as part of the scope of their work. (Hold our supposition in memory—we return to it shortly.)

Irrespective of this, another generalization to Jackson's network appeared in 1975 written by F. P. Kelly. This research also admits multiple classes of customers; however, he represents the queueing disciplines in highly generalized manner [22: 542]. In the end, Kelly produces the same marginal state distributions for the disciplines considered in BCMP [11: 32]. Then again, Kelly's formulation achieves this by tracking the customer class in each position of the queue for all queues [22: 547]. BCMP present the marginal distributions using only the number of each class at a given queue—the state description for FCFS requires the same detail as Kelly's work, but not the marginal distribution [3: 254-256].

Nonetheless, Gross and Harris state that Kelly's results allow the customer classes to have different service time distributions at a queue when the queueing discipline is FCFS [17: 246]. This would mean an even greater generality in the study of networks, but it appears to contradict the supposition previously given in the BCMP findings.

While it now appears that we have the capability to analyze any network, the undertakings of Kelly "give a more theoretical exposition…[than] explicit results on state-distributions of practical interest [11: 32]." Additionally, Kelly's abstract contains a worrisome statement: "The type of a customer is allowed to influence his choice of path through the network, and, under certain conditions, his service time distribution at each queue. The model assumed will usually cause each service time distribution to be of a form related to the negative exponential distribution [22: 542]." However, Kelly does not clarify this, nor does the work provide the means to support the assertion of Gross and Harris.

Following Kelly's paper, two papers co-authored by Chandy provide an in-depth look at the concept of local balance and the product-form. The first paper introduces the additional concept of *station balance*, which relates the rate of flow into and out of a given position within an individual queue. This is a finer resolution than that of local balance (state changes of the queue), and also requires tracking the customer class in each position of the queue. Recalling the four queueing disciplines (FCFC, PS, PRLCFS, and IS), station balance is satisfied only if a new customer begins service immediately, which cannot occur with FCFS [8: 256]. This is an intermediate point, not the conclusion.

The principle result of the work is summarized best in their own words, "[The] product form exists if and only if either the discipline satisfies station balance or all service time distributions are negative exponential with the same mean [8: 257]." We already have that FCFS is not station balancing, and class-dependent service rates violate the second condition. Accordingly, we see that there is a basis for the supposition made regarding the BCMP paper.

An intervening paper by Noetzel places the BCMP disciplines (FCFC, PS, PRLCFS, and IS) into a category he calls last-batch-processor-sharing. These disciplines are of a super-class of *symmetric* queueing disciplines [34: 781]. Symmetry requires that all customers in the queue receive equal shares of the server's time. What we learn from this work is that the product-form exists when the queueing discipline is symmetric. It appears that Noetzel purports that classes must have the same exponentially distributed service times under FCFS to satisfy his requirement of symmetry [34: 791].

Following Noetzel's work, Chandy co-authored a second paper on the product-form. Here, they derive the state balance equations using a differential equation approach they credit to a 1978 work by Kobayashi [10: 287]. This work contains the same conclusion: Classes must have the same exponentially distributed service times under FCFS to have a product-form solution [10: 297].

Is there an alternative to the product-form solution? Recalling an earlier discussion, we mentioned that BCMP discusses the concept of balance. In addition, we pointed out that balance is necessary to a solution. One alternative is to solve the balance equations numerically to obtain the state distributions, hence congestion measures. Another method is to use recursive techniques to obtain probabilities for a few states, and then obtain queue length distributions [9: 287]. However, necessary to these solutions is stationary equilibrium, which means that the system state distribution converges to a limiting distribution (i.e., time invariant).

There is another method called *Operational Analysis* published by Denning and Buzen in 1978 [12]. This method obtains performance measures from a practical standpoint using directly measurable quantities instead of fitted distributions. For example, the arrival rate is the number of arrivals during a *particular* observation period divided by the length of the period. While this sounds like the classical definition, the quantity has a single value specific to each observation period. Quantities obtained in this manner are then used to give familiar congestion measures (e.g., utilization).

The motivation for this work derives from the simple fact that "an analyst can never be certain that an equation derived from a stochastic model can be correctly applied to the observable behavior of a real system [12: 227]." In other words, a system does not always reach stationary equilibrium. The method is given to have three major areas of application: performance calculation, consistency checking, and performance prediction. However, the focus is on a specific period, as opposed to a general characterization of a network. Lazowska et al provides a complete text on this topic [30].

We close this section by providing the assumptions embedded in the product-form and the conditions necessary to a solution. This is the *Product-Form Theorem* as given by Chandy and Martin (shown on next page). Also, many texts take the time to illustrate the steps of finding a solution using the product-form. Some good examples using Jackson's construct are Kobayashi [24: 161-167], Schwartz [41: 219-222], and Medhi [32: 237-244]. Medhi spends considerable effort on illustration of the balance equations, global and independent, as well as local.

Product-Form Theorem [**10**: 296-297]

Framework Assumptions:

(1) The service discipline depends on only on the number of customers of each distinguished class.

(2) Arrivals from outside the network are generated by a Poisson process.

(3) The routing probabilities…are constant, independent of system loading.

(4) The service time distributions are differentiable.

Product-Form Theorem: For a queueing system in the framework given above, the steady state [probability distribution function] for that system satisfies product form…if and only if the discipline is balanced and for each distinguished class-k queue either

(i) the service times of all customers of that distinguished class have the same exponential density, or

(ii) the discipline is station balancing for distinguished class k,

or both (i) and (ii) hold…

FCFS is not a station-balancing discipline and so cannot have product form with nonexponential service times.

Approximation Techniques

Though not discussed prior to now, the results of the previous section depend upon the concepts of Markov chains, embedded Markov processes, and reversibility [47: 545-560]. Without belaboring the point, a Markov process has the property that the next state of the system depends only upon its current state, not the past [39: 304]. Reversibility says that the system has the same joint distribution for the state when the process operates in reverse time [27: 312]. This is a sufficient condition for the existence of the product form solutions. Also included here is the concept of stationary equilibrium. We point these out for additional insight, and as a reminder that analytic solutions are geared toward the long-run, or equilibrium, state of a particular system.

When constructing networks, the use of Poisson arrivals and identically distributed exponential service times is an example of a *Markovian* queueing network. Networks of this type create independence among the queue nodes crucial to the success of the product-form [39: 546]. For this example, this owes to the *memoryless* property of the exponential distribution, which permits independence from the past events, or states [28: 45-47; or any suitable Statistics text].

However, if we allow generally distributed service times, or even exponential with class-dependent means, at a FCFS queue, we destroy the Markovian property and no longer have a product-form solution. This is so because the remaining service time of a customer in service impacts the waiting time of a queued customer in a manner that cannot be accounted for in the product-form [40: 75]. Chandy and Sauer provide a similar version of this discussion [9: 281-289].

To contend with this, we turn to approximation methods to estimate the congestion measures. Generally, these fall into two categories: decomposition (aggregation) and diffusion. The decomposition approach views the network as a set of isolated queues—like an extended product-form solution, but with fewer restrictions. Diffusion, on the other hand, models the network analogous to a physical process having a continuous flow (e.g. an electrical circuit or a river) [32: 382].

Decomposition. In 1976, Kühn published what appears to be the first comprehensive work on an algorithm for approximation by decomposition [25]. In reality, this may be the second work, but we defer this point to later. The objective of his effort is to provide an approximation method for Jackson's network with general arrival and general service distributions. Also, the technique allows arbitrary queueing disciplines; however, only one customer class can exist.

To accomplish this generalization, you decompose a network into its component pieces (i.e., the indvidual queue/server pairs). The purpose of this decomposition is to apply analytic equations obtained for queues operating in isolation to each piece of the original network (for example, see [17], who discusses many exact and approximate results for isolated queues). One then aggregates the results of these individual calculations to obtain the overall network performance.

However, the queues do not operate in isolation—the departures from one may actually become arrivals to another. Relating findings of his peers [25: 236-1 – 236-2], Kühn points out that product-form networks decompose exactly under particular assumptions concerning the arrival processes. Indeed, that was Jackson's principal finding. Thus, Kühn's approach was to apply decomposition to general networks through special consideration of the arrival and departure processes at each node.

Appropriate representation of the arrival and departure processes is the heart of this technique [25: 236-2]. This is because these processes are the interactions between the queue nodes. Yet, the model looks at each queue as operating in isolation. Thus, a necessary assumption is that you can represent the arrival process to each queue as an independent, stationary *renewal* process.

Definition-wise, "If the [time between arrivals] are statistically independent random variables, each of which has the same probability distribution function, then the arrival process is called a *renewal process* [28: 46]." All we are actually saying here is that the arrival stream need not be Poisson to be a renewal process. For the strong-willed, Chapter 8-9 of Kulkarni [27] and Chapter 7 of Ross [39] discuss this in-depth.

From R. L. Disney et al [13], we have that, for a serially connected network, little reason exists to reject assuming a renewal process for the arrivals to each queue. However, if a customer may immediately return to the queue from which it just completed service, then the arrival stream is no longer independent and the process is not renewal. That is to say, unless all arrivals are Poisson and all service times are exponential, which maintains the renewal property. (Note: Kühn could not possibly know of the work we have cited because of its date; however, he indicates awareness of the problem.)

This re-circulation is a situation known as *feedback*, and it poses problems for the renewal assumption [25: 236-4]. One solution is elegant in its simplicity: eliminate feedback. Under feedback, the expected number of visits by a customer follows the geometric distribution. Thus, by giving each customer the total expected service time on the first pass through the queue, you can eliminate the feedback path.

Although you eliminate feedback, the original process is still non-renewal. What you gain is a masking effect that helps to produce a better estimation. Kühn relates that this is based on a proof given by L. Tackás in a 1963 article [25: 236-8]. For additional assurance, Kühn notes that this modification produces results closer to simulation values than without the modification.

Now that we have covered the renewal assumption, we reach the second major assumption: that you only need two parameters to represent the arrival, departure, and service processes. Actually, this is the third assumption since the first is that you can apply the decomposition method to a general network. The idea here is that the mean and the squared coefficient of variation (SCV, variance divided by square of the mean) contain enough information to represent the processes.

Using this assumption, Kühn now has the means to decompose the network, while retaining information concerning the interactions. This is information comes from an equation relating the departure SCV to the arrival and service SCV given by K. T. Marshall in 1968 [31]. Additionally, this assumption allows Kühn to apply an approximation due to Krämer and Langenbach-Belz for the expected wait at a queue [25: 236-2 – 236-3].

Summarizing up to this point, we first assume that you can decompose a general, non-Markovian, network (that is why this is an approximation). Next, we separate the queues and say that the arrival process to each queue shall be renewal. If feedback exists, which causes the process to be non-renewal, take steps to eliminate the feedback. Then, obtain the departure SCVs and solve for the arrival process to each queue. Finally, estimate the congestion measures using other approximation equations.

Without discussing how one actually calculates congestion estimates, the previous encapsulates the major aspects of Kühn's method. To illustrate the suitability of the assumptions of this method, Kühn provides several graphical comparisons to simulation output [25: 236-5 – 236-7]. As an extension, Kühn suggests adding customer classes; however, he envisions that each queue node must actually contain a separate waiting line for each class [25: 236-7].

Though long in presentation, we later need this understanding of Kühn's work. In 1979, Kühn published the work in the US under the surname of Kuehn and with a minor change in the title [26]. The only significant difference is the removal of some material from the appendix and addition to the bibliography. Oddly, Kühn fails to reference the 1976 version in any manner. He does improve notation, but the formulations remain unchanged. Disappointingly, he does not provide additional consideration for the case of multiple customer classes.

The next stop along this route is to a paper published in 1982 by W. Whitt [48] and to a companion paper also in-work in 1982, but not published until 1984 by S. Albin [1]. The purpose of Whitt's paper is to provide a comparison between two methods of approximating an arrival process by use of renewal processes. Then, Albin's effort combines the two methods as a convex combination to provide a better approximation tool.

Before discussing Whitt's two methods, we need to augment our discussions on Kühn's approximation of the arrival process. What we implied, but never really stated, is that the arrival process to a queue may be the sum of external arrivals, as well as the departures from any queue. We already noted feedback that occurs immediately, but we did not discuss feedback delayed by loops or circuits within the network, which also creates dependence. Thus, Kühn's actual assumption is that the superposition (sum) of these arrival streams, under the possible presence of delayed feedback, gives an independent renewal process at each queue [25: 236-3 – 236-4].

Now, Whitt notes that Kühn's approach is an application of the *stationary interval* method [48: 126-127]. In this, you match the renewal approximation to the system behavior using a short time interval. However, the renewal intervals in Kühn's approximation are from a superposition process, which are not independent intervals. In 1972 E. Çinlar showed that, in order to have independence, the separate streams themselves must be Poisson for the superposition to be renewal [48: 138]. Therefore, the stationary interval method may not always provide the best results since you are looking at short interval in a process with dependencies.

In order to account for the dependence, Whitt suggests exploring the *asymptotic* method [48: 125, 127]. In this method, you match the approximation using a large time interval. This actually means that you match the approximation using the asymptotic behavior over sums of successive intervals.

How does one matches the approximations to the process? We again use only the mean and the SCV. Earlier, we discussed Kühn's rationale for selecting this approach; however, Whitt gives a more compelling reason. Namely, if you try to use more parameters to characterize the distribution of the arrival process (higher moments), then it is possible to have inconsistency in these additional parameters [48: 134-135]. This inconsistency comes from approximating a renewal process by a superposition.

Getting to the comparison, Whitt uses simulation to obtain estimates of error for the two methods separately [48: 143]. He indicates that for some scenarios, which he does not list, neither method performed well. However, when the SCV of the superposition is greater than one, the asymptotic method tends to overestimate the mean queue length, whereas the stationary interval method tends to underestimate. When the SCV is less than one, the two method reverse in estimation.

In addition to those observations, Whitt relates some known properties of the two methods[48: 128]. From Çinlar (1972) and Khintchine (1960), the stationary interval method is asymptotically correct as the number of arrival streams gets large. And, from Whitt's own work in 1979, the asymptotic method is asymptotically correct as the server utilization, or traffic intensity, approaches one.

The above comparison and observations lead Whitt to the belief that a convex combination (hybrid approximation) of the two methods would work well. Here, Albin notes that the accuracy of the hybrid depends on the distribution type of processes you merge [1: 1153]. After developing the weighting scheme, she uses six different distributions sets to validate the approximation: 1) hyperexponential with balanced means, 2) lognormal, 3) half hyperexponential and half lognormal, 4) half exponential shifted by a constant and half gamma, 5) exponential shifted by a constant, and 6) gamma. However, she acknowledges that she could not assess quality under different arrival rates and different distributions. Notwithstanding, she gives that the average absolute error of the hybrid is less than 5 percent [1: 1137].

Let us take a moment to reconsider what Whitt suggests. Earlier, we gave that the asymptotic method is asymptotically correct as the utilization, or traffic intensity, approaches one. And, relating to his 1979 work, Whitt notes that the asymptotic method is the heavy-traffic limit of the equivalent random method [48: 129]. Consequently, we see that Whitt recommends blending a decomposition approximation with a diffusion-like approximation (a decomposition approximation using asymptotic results).

In 1983, Whitt does this in a paper entitled "The Queueing Network Analyzer" (QNA). This paper describes a parametric-decomposition method that blends Kühn's efforts, the results of other's efforts (see [49]), and the hybrid approximation [49: 2783]. The result is a software tool built to provide approximate analysis of non-Markovian queueing networks. To our benefit, we discussed practically every aspect of QNA in getting to this point. Nevertheless, there are a few additional notes to make.

First, Whitt is quick to note that the heuristic approximation of the asymptotic method is not the same as heavy-traffic limit theorem and actual diffusion approximations—those are more complicated [49: 2783]. However, Whitt uses 16 pages to explain this and present the related, non-intuitive equations [49: 2792-2807]. Needless to say, we refer the reader to the published work.

Earlier, we noted that Kühn used an approximation due to Krämer and Langenbach-Belz for the expected wait at a queue. QNA supplements this by explicitly using the form due to K. T. Marshall [31]. In actuality, they differ only by an extra term, which Marshall omits. However, Whitt notes that the extra term reduces accuracy when the arrival SCV is less than one [49: 2802].

While QNA permits most of the generalities Kühn allows, it does take one step back. Specifically, QNA only handles the FCFS queueing discipline. Though Whitt does not cite direct causality, we find that the traffic equations (equations that relate the arrival and departure processes) in QNA have a simpler form. However, this does limit QNA's scope of applicability.

Here is a small bonus: QNA does permit multiple customer classes, and each class may have its own service time distribution at a given queue. Furthermore, the distribution can be different for each visit to a particular queue. However, the class routing is deterministic, and the user must specify the complete path of each class [49: 2789].

As before, we do not want to go into the particulars of calculating estimates of the congestion measures. The procedure is as before: decompose the network, eliminate feedback, estimate the arrival process to each queue, and aggregate the results. The primary difference here is that this is done using systems of linear equations [49: 2786]. To eliminate the nonlinear terms in Kühn's application of the stationary interval method, Whitt makes a further assumption that the superposition is a Poisson process [49: 2797-2798].

In a separate 1983 paper, Whitt presents the performance of QNA. Referring to Kühn's 9-node example, the results are indistinguishable from the graphical representation [50: 2833-2834]. However, Whitt notes that, for situations where a queue accepts arrivals from another queue and sends its departure to that same queue, QNA may yield incorrect results [50: 2829]. The interested reader should consult the actual paper to get a better feel for the numerous cases presented.

Although QNA was written as a software product, obtaining a copy appears difficult if not impossible. While the above papers discuss the first version, it is known that Purdue University holds version 3.0 . However, AT&T Laboratories declined our request to obtain a copy from Purdue [52].

Up to now, we concentrated our discussion on decomposition methods. However, we did not explicitly discuss three efforts, but we wish to note them for documentation purposes. Specifically, we omitted works by Sevcik et al [43], another by Chandy and Sauer [9], and one by Shanthikumar and Buzacott [44]. However, we do not feel any information is lost since these works take us in the direction of increasing complexity (i.e., integrals and nonlinear equations), or are contained in [1; 48; 49].

Diffusion. Prior to the works of Kühn, those mentioned in the previous paragraph, and Whitt; Reiser and Kobayashi present a diffusion approximation for queueing networks in 1974 [38]. Though given as a diffusion approximation, this work also uses the two-parameter assumption (mean and SCV contain adequate information to represent the process). Given Whitt's admonishment, we see that this could be the first work on the decomposition approximations. But, in the 1976 paper, it is not clear that Kühn is aware of this work. Given that Reiser and Kobayashi's title is "Accuracy of the Diffusion Approximation for Some Queueing Systems," we will simply say that the two approaches developed in parallel.

We already described the diffusion approximation as a fluid-type model used when the utilization, or traffic intensity, approaches one. Actually, the expression for the expected waiting time at a queue derived from the heavy-traffic limit theorem is the strict upper bound when arrivals and services are general [32: 377]. In other words, we see that this category of approximations represent an extreme side. Decomposition methods, on the other hand, tend to be middle-of-the-road with respect to upper and lower bounds on the expected waiting time for general queueing networks.

Multi-class constructs also exist for the diffusion approximations. Gelenbe and Mitrani present a discussion on possible extensions in 1980 [15]. In 1993, Kelly and Laws published a work on the virtues of dynamic routing over deterministic routing, which QNA uses. In their use of dynamic routing, the customer chooses its route based on conditions in the network. However, they indicate additional work is required to support generalization [23: 84].

In competition with QNA, we found software by Harrison and Nguyen called QNET. This software is self-described as an *approximating Brownian system model* that is better than QNA [18: 1-2]. However, three years afterward in 1993, a second paper by Harrison and Nguyen appeared indicating that the "QNET approximation scheme is generally not valid [19: 6]." They note that mutli-class, FCFS queueing networks with feedback present the majority of the problems for their formulation [19: 7].

After some revisions to the QNET approximation, two numerical examples show the average error to be under 10 percent for determining the distribution of the system response time [19: 33-39]. However, graphical illustrations indicate that convergence to simulation values of response time does not occur until traffic intensity exceeds 0.80 [19: 38-39]. The status of this particular method and the software are unknown.

Because of the upper-limit nature, diffusion methods come with their own class of problems. In general, you cannot obtain information about queues in light traffic because the waiting times are set to zero [16]. Because of this, there may be large errors in the remaining estimates [28: 218]. Browne and Whitt present a more enlightened and optimistic view on the use of diffusion approximations. In particular, they believe that diffusion processes are elementary and should be the first candidate in any queueing application [5: 463].

Summary. Reviewing this section, for the decomposition methods, we first assume that you can decompose a general, non-Markovian, network. Then, you separate the queues and say that the arrival process to each queue shall be renewal. If feedback exists, which causes the process to be non-renewal, take steps to eliminate the feedback. Then, obtain the departure SCVs and solve for the arrival process to each queue. After all that, estimate the congestion measures using other approximation equations.

However, if all nodes in the network are under heavy traffic, a diffusion approximation may produce better results. These methods view the network as a continuum, rather than isolated queues. But, in some cases, the steps in a diffusion algorithm parallel the decomposition method, which avoids some of the complicated mathematics.

Multiple Customer Class Considerations

In the approximation methods just covered, we found that they admitted multiple customer classes under some restrictive rules. For example, QNA requires complete specification of the itinerary of each class. Kelly and Laws discuss dynamic routing, but there are some implementation issues. Lacking in the works found was any mention of probabilistic routing.

Regardless of the routing method, we did see a consistency in how each represented the flow of customers between nodes. Specifically, we saw prevalence of the two-parameter method: use the mean and the squared coefficient of variation (SCV, variance divided by square of the mean) to represent arrival, service, and departure processes. This was presented by K. T. Marshall in 1968 [31], with some changes allowed in later forms. However, there seems to be a general dilemma for multiple classes.

The concentrated study of output processes actually began around 1956 with Burke's analysis of two queues in series [6]. Though there were some slight problems [7], the work inspired Jackson's effort, which eventually lead to the decomposition algorithm by Kühn.

Regarding the assumption that arrivals to a queue node can be assumed renewal for a network of arbitrarily connected queues, significant disquiet existed as to whether or not one should make this assumption [13]. Addressing Kühn's handling of immediate feedback, Disney et al generally accept the technique, but note without being specific that it would not be acceptable for determining anything other than queue lengths [13: 637-638].

Having the benefit of hindsight, we are not terribly concerned with that specific observation. However, we are concerned with the restatement of Tackás' proof. In illustrating why the assumption is acceptable, Disney et al state, "…since customers are indistinguishable…[13: 637]." Clearly, if one plans to include multiple classes, this could be an area of extreme concern—the classes are not indistinguishable.

Previously, we also talked to difficulties related to issues of independence in the superposition of feedback, in general. We simply noted that independence is lost when a customer returns to a previously visited node. Melamed gives an expanded explanation for the case of multiple classes. Basically, if one customer class can get in line ahead of (pass) another customer class that is returning for additional service, then a dependency exists that is difficult to accommodate [33: 223].

In a work that predates QNA, Melamed presents a multi-class decomposition method that allows different service distributions. However, he accomplishes this by *conditioning on the customers' itinerary* [33: 226]. This sounds similar to the method used to allow classes in QNA. Additionally, the method is such that it excludes FCFS because of the possibility of passing [33: 239]. Indeed, we omitted several works because of this restriction on passing.

Returning to our general discussion on superposition, we offer one additional observation. Disney and Kiessler, in a text devoted entirely to the study of traffic process, comment on the simplifying assumption in QNA (though not aimed specifically at QNA) that leads to the linear form of the hybrid approximation. They say, "…the conclusion that the nodes in these networks have Poisson input processes is not justifiable. Nor can we conclude that the several input processes to one node or to any collection of nodes are independent processes [14: 6]." Furthermore, "...these networks with loops and with FCFS discipline never have renewal departures except in the case of exponentially distributed service times [14: 7].

Do not construe this as a specific criticism of QNA, but as a criticism of approximation methods in general. What we are getting at here is that significant theory weighs against including multiple customer classes in a general framework.

Regardless of these observations, Whitt published "Towards Better Multi-Class Parametric-Decomposition Approximations for Open Queueing Networks" in 1994 [51]. This method attempts to extend Marshall's 1968 result [31] to include multiple classes. Without mincing words, "Methods are developed for approximately characterizing the departure process of each customer class from a multi-class single-server queue with unlimited waiting space and the first-in-first-out service discipline. The model is [general] with a non-Poisson renewal arrival process and a non-exponential service-time distribution for each class. The methods provide a basis for improving parametric-decomposition approximations for analyzing non-Markovian open queuing networks with multiple classes [51: 221]."

To the best of our knowledge, this presents a reasonable portrayal of the past. In addition, to the best of our ability, faithfully represents the current state of affairs.

III. Development and Findings

Here we present the incremental findings towards satisfying the objective. After briefly reviewing the objective and scope, we give an expanded statement of the general methodology. Following that, we begin a review of significant findings for each task. We combine the development of any necessary relationships with the review. At the end of the chapter, we show how one could apply the results of this effort to focus a simulation study.

Review of Objective

The objective of this thesis is to increase the flexibility of approximation techniques in obtaining estimates of congestion measures for complex, open queueing networks. That is, to easily obtain these estimates, before building an intricate simulation model, for networks having several customer classes and class-dependent structures.

That sounds great, but let us look a little deeper into the motivation before going further. For large network models having a complex solution or without an exact solution, simulation is likely to be the best tool to characterize the general behavior. If you have simulation software with a graphical interface, it is generally easy to quickly create a model and obtain summary statistics. Of course, this is relative to the complexity of the model.

However, ease of use diminishes as you add class-dependent structures (e.g., service times) and endeavor to obtain more detailed statistics (e.g., the actual waiting time of each class at a queue instead of the aggregate time). This decrease is the result of complexity due to additional overhead the user must incorporate to handle class dependencies. More overhead means more time spent building the simulation model. Plus, you have increased opportunity for mistakes as the level of detail increases.

What if you are only interested in adding that level of detail for queues meeting some criteria, like longest average queue length or shortest wait based on customer class? This modeling approach would certainly decrease the amount of time spent building the simulation model, but structure of the network model may not readily indicate where this will occur. What one needs is a tool that shows where to focus effort in constructing the simulation model. This takes us back to the stated objective.

Summary of Scope

We conduct the study on an open network having a finite number of arbitrarily connected queues. The external arrivals consist of different classes of customers that may have different service time distributions at any queue. In addition, all queues are single-server, first-come-first-serve (FCFS) with unlimited capacity. Routing between the queues is based on specified branching probabilities, and customers may visit a node any number of times. For this work, entry to the network is restricted to a single node, but customers may depart the network after completing service at any node. Figures 1 and 3, introduced later, schematically illustrate different realizations of this description.

This effort seeks to find a procedure to estimate performance of a network involving multiple customer classes. Or, if a procedure is not readily apparent, provide groundwork on development towards that end. To make the task manageable, we estimate long-run congestion measures. We do not address transient (startup) behavior.

Overview of Methodology

- Conduct literature review of exact solution techniques to verify inability to accommodate the objectives of this effort.
- Conduct literature review of approximation techniques to determine state of current research (i.e., has someone solved this problem).
- Obtain an existing, or develop a new, approximation algorithm for queueing networks.
- Construct a queueing network model having an exact analytic solution (base case).
- Construct and verify a simulation model of the base case.
- Verify the estimates of the approximation algorithm for the base case.
- Introduce class-dependent service rates in the simulation model.
- Observe the congruency of the approximation algorithm and note any failures.
- Increase the complexity of the network model, obtain estimates from the approximation algorithm, and check agreement with simulation model.
- Illustrate use of approximation as a precursor to simulation.

Highlights of Literature Review

The class of queueing networks we desire to analyze does not have an exact analytic solution. The works co-authored by Chandy [8; 10; 40] repeatedly show that networks containing FCFS queues with class-dependent service time distributions are intractable. If this is not true and a solution actually does exist, the current state of queueing theory is not advanced enough to reveal the form.

It is generally true that one can make simplifying assumptions that allow you to use an exact solution. However, this may be undesirable since you must alter the network to fit the framework of an exact solution. The decision as to whether or not this is acceptable is left to the user. In our case, the only possible simplification appears to be forcing equal service times under the exponential distribution. But, which is better, an exact solution to an approximate model, or an approximate solution to a more exact model [49]? This lead us to the approximation techniques.

For cases where the servers of the entire network are continuously busy, diffusion approximations seem appropriate. These techniques may be suitable if you only need to identify bottleneck locations. However, we choose to exclude heavy-traffic situations from this research effort.

The greatest flexibility comes with decomposition methods, where the network is viewed and analyzed as being composed of isolated queues. There appear to be several proposed techniques; however, all seem to be variations on a theme with no one being clearly better than another in terms of accuracy.

Selection of Approximation Algorithm

We decided to use the algorithm of the Queueing Network Analyzer (QNA) for this effort. This choice is based upon ease of implementation in a computational environment. During the review, we found that most published works generalized the content beyond the point of being able to readily construct a working application from them. However, we found that the paper on QNA [49] contains enough information to reconstruct the algorithm rapidly in a symbolic environment (e.g., Mathcad®), or code the method in your favorite high-level language (e.g., FORTRAN). Additionally, it is easy for the user to modify the algorithm to test different assumptions. For example, you can remove the existing equations that relate arrivals to one queue to departures from the another, and replace them with other approximations. Thus, the QNA algorithm does not bind you to its internal approximations, as long as your new approximations are linear.

Again, QNA is not the only answer, but it does eliminate the need for us to attempt building an approximation from scratch. There are commercial applications created to analyze communication networks; although, we did not use them for this effort. It may be possible to modify these for our needs, but they appear too specialized for the general network studied here.

If you recall, QNA does support multiple customer classes, although, the implementation is more restrictive since you must prescribe routes for each class. Thus, our goal is to further generalize this decomposition algorithm with probabilistic routing.

A Trial-Size Network (Base Case)

Selecting a Network. Once we add class-dependent structures to our decomposition algorithm, we need to know if it produces reasonable congestion measure estimates. We will use simulation to provide comparisons since the scenario is analytically intractable. Therefore, the simulation becomes our indicator of the *true* state after introducing class-dependent structures.

To anchor the validity of the comparisons, we start at a common reference point—a base case network with a known solution. This allows us to determine whether we reconstructed the approximation algorithm correctly, and provides a check for estimation biases in our simulation.

Figure 1 shows the base case network. The external arrivals are Poisson at rate λ_o and enter at node 1. By superposition of Poisson processes, this arrival stream can represent any number of classes. Node 1 represents an FCFS queue with a single server working at rate μ_1 (later we use only a circle to represent this combination). For the base case, the service time distributions are exponential with the same mean for all classes. From node 1, customers depart the network with probability p_{10}, or proceed to node 2 with probability p_{12}. Node 2 is the same as node 1, but with service rate μ_2. All customers completing service at node 2 return immediately to node 1. The returning stream must merge with the external arrival stream—this is delayed feedback.

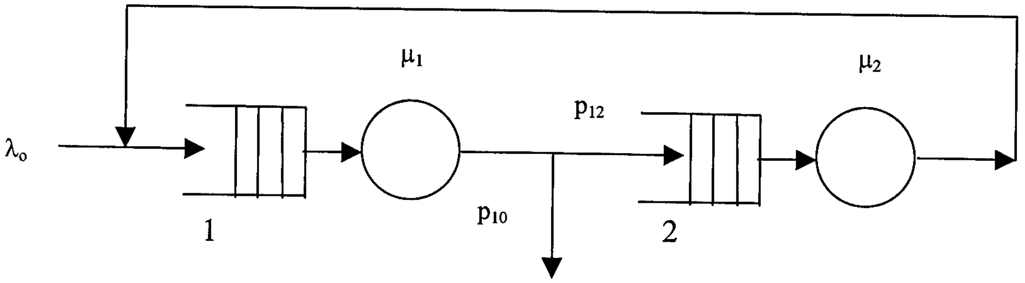


Figure 1. Two-Node Network [46: 412]

As long as all classes have the same service time distribution at a given node, this network has a product-form solution. Letting n_i denoted the number in the queue at node i, we have that the joint probability distribution is given by

$$p(n_1,n_2) = (1-\rho_1)\rho_1{}^{n_1} \bullet (1-\rho_2)\rho_2{}^{n_2} \quad , \qquad (1)$$

where $\rho_i = \lambda_i / \mu_i$ is the traffic intensity at node i (same as server utilization).

Using this equation and all possible values for the numbers at each queue, we can calculate the congestion measures. Pursuing the solution in that fashion is difficult even with this small example. Instead, using results in examples given by Trivedi [46: 412-417], we obtain equations that allow us to calculate the congestion measures. As an example, the expected response time (average time spent in the network) is given by

$$E[R]=\left(\frac{\rho_1}{1-\rho_1}+\frac{\rho_2}{1-\rho_2}\right)\bullet\frac{1}{\lambda_o} \quad . \tag{2}$$

Setting Up a Scenario. For our base case scenario, we begin with two customer classes arriving to node 1 at different rates. Keep in mind that we do not distinguish between the two classes in terms of their service requirements at this time, but we want the architecture in place for later use. For class 1, $\lambda_{c1} = 1/10$, and for class 2, $\lambda_{c2} = 1/15$. From the theory of superposition, the external arrival rate, λ_o, is just the sum of these two since both are Poisson. We selected unequal service rates for the queue nodes. Node 1 has $\mu_1 = 2$, and node 2 has $\mu_2 = 3$. This means that node 2 processes slightly faster than node 1. Then, we set the probability that a customer departs the network after completing service a node 1 at $p_{10} = 0.30$.

Appendix A illustrates our solution for this network. We will refer to this as our *exact* solution, and we remind the reader that this is so only as long as the arrivals are Poisson and the service times have class-**in**dependent exponential distributions. After some math to calculate the internal arrival rates, we get that the traffic intensity at node 1 is $\rho_1 = 0.2778$, and is $\rho_2 = 0.1296$ at node 2. This gives an expected response time of 3.2013 units; we show the other results later.

Simulation Model for the Base Case

Model Building. After choosing our base case and obtaining the solution, we create a *high-resolution* simulation model and then *calibrate* the simulation performance against the base case. We want to include all simulation features needed to track multiple customer classes (high resolution) before running the model to match the base case solution (calibrate). This is because seemingly minor changes to the simulation model, later, may cause changes in the results not related to the customer classes.

To put the complexity differences in perspective, note that Figure 1 contains two components (server and queue count as one) and four arcs. The simulation model (shown in Appendix B) contains 131 components and 172 arcs to represent the **same** network and account for the movements of **only two** customer classes. Granted, there are structures in our simulation model (90 components and 41 arcs) that you only need for this study (e.g., arrival and departure time intervals). However, you need the remainder to capture congestion measure information by customer class. Building in all overhead at the beginning allows us to check the model coding so we can develop our ideas and test them using a tool that we verified as properly built.

We will not present a tutorial on our simulation software (AweSim®, [37]), but we should briefly describe the structure of the model. The next paragraph generally describes the schematic, and then we return to the discussion. Ignore the specifics of the schematic in Appendix B, but get a flavor for how the simulation model size can vary based on the level of information you desire.

Figure 2 shows two of the symbols used in the simulation model. Every node shown on the simulation schematic has a rectangular tag giving the node name, and the names are descriptive where important. The model begins at two *create* nodes for two types of customer having some time between creation that is exponentially distributed, which is held in an attribute. After creation, each class is *assigned* their birth type before passing through a series of nodes that *collect* information on the arrival process. Just before reaching a *queue* node, each class is assigned its service time for the given queue. We do this in this manner so that you can differentiate between classes. Later, we collect service time information, which occurs immediately after the server in another series of nodes. The arc leaving the queue node represents the server. After service at the first queue, we leave the network or go on to the second queue based on some probability.

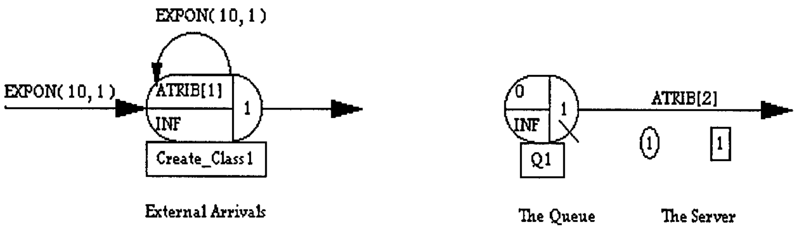


Figure 2. Simulation Representation of Nodes

Output Analysis. Next, we determine if our simulation output matches the exact analytical results. This involves application of *output analysis* covered in simulation texts and literature [2; 29]. We need to properly consider this area, but without turning this into a simulation study. The question is: What do we need to do to be comfortable with the simulation output?

For our output analysis, we first consider that the estimates are for long run performance so we do not have an identifiable stopping point in time. The alternative solution to this is to have a stopping point based on the number of departures from the system. This led us to look at departures ranging from 10,000 to 200,000 for comparison against the base case.

To assign statistical significance to the simulation output, we consider conducting multiple replications and average the means of each run, or parsing the output from one long run into sections (*batch means*) and average the batches [2; 29]. We found that short runs tend to be biased, and we need upwards of 25 replications to get reasonable estimates with low standard error. A single long run converges to the same mean performance as the replications; however, we found that the logical batch points (where all servers are idle) are not regularly spaced. This creates some difficulty because we need to parse the run so that the batch means are not correlated. After attempting to batch a very short run (1000 departures), we decided that the method was not practical here.

For the class-dependent work later, we just want to run the simulation to a point where the means resemble steady state values. We use only a single long run for these comparisons because we also want the variability estimates from the run itself. This is a valid as long as the system is not in heavy traffic, but we will not calculate a confidence interval for measures from these runs. For instances where we need to substantiate a line of reasoning, we use replication of the shorter runs to estimate a confidence interval for the means.

Referring back to the base case discussed in the last section, we found the expected response time for the base case was 3.2013 units. For the simulation, we found that we could only obtain 3.185 (no confidence interval, we are only looking at the observed mean). Believing this to be initialization bias caused by starting from an empty and idle condition, we attempted to correct the problem. We looked at deleting the startup period, starting the run with the queues containing the expected number of customers (actually, just over the expected number), and a combination of both. Since we are not under heavy traffic, no method works well because the system often returns to an empty and idle condition. The conclusion here is that we need to run the simulation beyond 200,000 departures and possibly alter the starting conditions to reduce the bias.

Using the technique of *ensemble* averages (replicate long runs, batch each run with equal batch intervals, and average batches across replications), we might obtain an estimate of how much longer we need to make the run and how to account for the initialization bias. However, doing so becomes self-defeating at this stage because the time required overshadows any benefit derived from forcing the estimates to be exactly equal. Again, we stop at 200,000 departures for most comparisons and acknowledge the apparent bias. At the end of the next section, we have one more point to make regarding bias.

After that discussion, one might point out that we could replicate the long runs. There are two problems with this: run time and length of random number streams. For this simple, two-node network, a Pentium II™ 200 requires approximately 18 minutes to process a single replication with 200,000 departures. During each replication of the base case, the first queue node will process on the order of 660,000 entities and the second will process approximately 460,000 entities. Although we attempted to space the random number streams to allow replication, we run the risk of overlapping both of the service time and the branching streams after three replications.

We want to present all three estimates of congestion at the same time. The next section contains the base case congestion measures for the exact solution, the simulation, and QNA. At this point, we note that the simulation has good **face-validity**.

Comparison of Estimates for the Base Case

The previous section discussed the effort directed toward verifying that we constructed the simulation model correctly. Additionally, since we specified the base case model that we are trying to embody with our simulation model, we were also validating that we constructed the correct simulation model. This section presents the results of that effort below. Additionally, we extend the validation effort by discussing and illustrating two problem areas that hold in general beyond the base case.

Before we present the comparison, we quickly discuss building QNA. Recreating QNA at the level of a simple tool is straightforward using the published work [49]. In fact, we basically copied the equations directly into Mathcad®, but we did not include the variability estimates given in the paper. By simple, we mean that the reconstruction does not provide the error checking and automatic reconfiguration (eliminate feedback) features of the actual software [42]. It is possible that using the actual code could have provided additional insight to our task. However, obtaining a copy of the code is difficult since this was not production software

Verification and Validation Results. Table 1 contains a summary of the results obtained for the base case after determining the most reasonable operating conditions for the simulation. This is a comparison between all three methods. For the simulation, we show the averages for 25 replications of a short run (10,000 departures) and a single long run (200,000 departures). The abbreviations are as follows: EN_i for the expected number at node i, EW_i for the expected wait at node i, and ER for the expected response time of the network.

Table 1: Base Case Results

Congestion Measures	Exact	Simulation			QNA
		10K Depart, 25 Reps		200K Depart	
	Mean	Mean	Std Error	Mean	Mean
EN_1	0.1068	0.106	0.001	0.106	0.1068
EW_1	0.1923	0.191	0.002	0.192	0.1923
EN_2	0.0193	0.019	0.000	0.019	0.0193
EW_2	0.0496	0.049	0.000	0.050	0.0496
ER	3.2013	3.185	0.012	3.185	3.2013

Note that individual 95 percent confidence intervals on the replication values capture the exact solution equivalent. Additionally, the single 200,000 departure simulation run converges to near equality with the replication. These results are sufficient to establish validity of the simulation model under the base case. Appendix C contains the AweSim output summarized in this table.

The exact solution and QNA match for the base case, as they should if we reconstructed the algorithm correctly [49]. Therefore, this also provides initial validation for QNA. Appendix D contains an example of how one can recreate QNA.

Feedback. Along with showing good agreement in the estimates, the structure of our base case network allows us to look at problems related to feedback. In the literature review, we included a discussion on immediate feedback, where the routing structure allows a customer to immediately re-queue at the same node after completing service. There we noted that you should restructure the network and remove this form of feedback to improve the quality of the estimation [49: 2792].

If your routing structure produces a subset of nodes resembling Figure 1, your network has delayed feedback. This situation degenerates to near-immediate feedback if the service rate at the second node is significantly higher than the first. Here, the nodes become *tightly coupled*, and customers return almost immediately to the first queue [50].

We checked our base case for consequences due to this by ranging the node 2 service time over values approaching zero and comparing the results from simulation and QNA. Recall that QNA and the exact solution are identical for the base case, regardless of the service rates. Additionally, the estimates for node 1 will remain unchanged because the nodes are mathematically independent. We found no significant impact—the bias in the simulation remains relatively constant, as it should in a Markovian network.

For such networks, which decompose exactly, the effect of feedback (delayed or immediate) will not likely be apparent. There may be some disagreement with the QNA estimates because of the starting conditions for the simulation and the run length. We see in Table 1 that 95 percent confidence intervals on the replication values capture the exact results. Ranging the level of feedback did not greatly affect the relative accuracy shown in Table 1, but we did note some change in the simulation performance.

In further examination of QNA and the simulation, we found that the actual problem with feedback lies in how arrival variability affects the queue lengths and response time. The exact solution is true for the long run, but the shorter-running simulation is sensitive to variability in the feedback stream. This also may lead to bias in the simulation, along with run length. Furthermore, for service distributions other than exponential, the exact solution no longer holds because dependencies exist in the network. Then, the arrival variability is not just an artifice of simulation sensitivity, rather it represents an actual coupling between service times and arrival rates at paired nodes.

Although QNA attempts to capture these dependencies in the non-exact case, the algorithm may tend to overestimate the number waiting at the nodes seeing near-immediate feedback. Whitt suggests a procedure to address this situation, but the results are mixed [50: 2827-2832]. For immediate feedback, you can restructure the network and eliminate possible discrepancies due to differences in arrival variability representation between QNA and the simulation.

Heavy Traffic. Returning to the base case, the impact of feedback is compounded when we simultaneously place the first queue in heavy traffic. In any situation with heavy traffic, simulation will tend to underestimate the expected number waiting at a congested node, as compared to QNA. This is because QNA uses the exact solution for the expected number waiting, $\rho^2 / (1- \rho)$, which goes to positive infinity as ρ goes to one. The above expression derives directly from the product-form (Equation (1)) using a modified geometric distribution [46: 368-369]. However, this is an asymptotic limit that we will not likely realize without excessively long simulations. Note that, for the non-exact case, QNA multiplies the term by measures of the service and arrival variability.

Results under Feedback and Heavy Traffic. We discuss feedback and heavy traffic together because we encounter similar difficulties: possible underestimation of the expected number waiting and response time given by simulation. Table 2 shows a comparison using $\mu_1 = 0.56$ and $\mu_2 = 35.0$, but with the same arrival rates used for Table 1. Here, the difference is primarily due to heavy traffic at node 1 ($\rho = 0.991$).

Table 2: Tightly-Coupled with Heavy Traffic

Congestion Measures	Simulation		QNA
	100K Depart, 25 Reps		
	Mean	Std Error	Mean
EN_1	91.805	6.470	124.0079
EW_1	165.314	11.804	223.2143
EN_2	0.000	0.000	0.0001
EW_2	0.000	0.000	0.0003
ER	556.252	39.383	750.0674

For the base case network, the difference between the simulation and exact solution will increase as we move toward heavy traffic at any queue node. Therefore, we avoid heavy traffic. Once we include class-dependent service times in QNA, we must also be wary of network structures that lead to tightly coupled nodes. Near-immediate feedback may affect the quality of the approximations; however, for the base case, it does not. This does not mean that you will not be able to use the technique for nodes that have long queue lengths (heavy traffic) or unresolved feedback. It means that we must remain aware of these issues. Even if we are not interested in asymptotic performance, note that the exact solution from QNA in Table 2 shows us the correct trend information.

Including Customer Classes, I

Introduction. The purpose behind the work that went into the base case was to gain a greater understanding of the internal processes and illustrate possible problem areas. Plus, it gave us confidence in the validity of the simulation model under light to moderate traffic. We found that the largest error was in the estimation of the number waiting at node 2. That was at 1.6 percent below the exact value; however, the other values were less than 1 percent below.

As we add class-dependent service rates to the simulation, we retain confidence in the output. The output will change, but that is because the service order of customers may shuffle and they may experience different delays. For example, if we assign the service rate at node 2 to be 4.5 for one class and 1.5 for the other, the average service rate at the node is still three if the probability of seeing each class is equal. However, the first class now takes more time, on average, to complete service. Since the external arrivals are exactly the same as in the base case, we have a possibility that a returning customer of the first class (feedback stream) now finds a different customer ahead of him. This will induce changes in the variability of the arrival processes and waiting times, but the mean values will remain essentially unchanged. Additionally, the service variance at the node will no longer be that of the exponential distribution, even if the individual service times are exponentially distributed for each class.

Possible Internal Modifications to QNA. The QNA approximation uses the assumption that the mean and squared coefficient of variation (SCV, variance divided by square of mean) contain enough information to represent the arrival, service, and departure processes. This information is used to relate the dependencies that exist within the network under general service distributions. For example, with c^2 being the SCV for arrival (subscript a) and service (subscript s), we have one of the approximation equations for the expected wait in the queue being

$$EW = \frac{1}{\mu} \bullet \left(\frac{\rho}{1-\rho}\right) \bullet \left(\frac{c_a^2 + c_s^2}{2}\right) \quad . \tag{3}$$

The node arrival SCV is based upon external and internal input streams, and the service SCV is based on the overall service. However, the SCV does not provide information regarding the class of customer. As you can see, both c^2 values are class-**in**dependent.

In QNA, you obtain the node arrival SCVs by solving a system of linear equations that relate node departures to arrival and service rates across the network. The simplest departure relation at the node level has the following form:

$$c_d^2 = \rho^2 c_s^2 + (1-\rho^2)c_a^2 \quad . \tag{4}$$

An equation similar to this, albeit more complex, is imbedded in the traffic variability equations of our reconstructed QNA (Appendix D).

Now, the task is to find a method that permits us to obtain information from equations like (3) using an equation like (4) that solves the traffic variability equations on a by-class basis. The work presented by Whitt [51] contains several alternatives to (4) for just this purpose. If you know the probability of one class at a node, p_1, one choice is to replace the departure SCV as a combination the arrival SCV of two classes.

$$c_a^2 = p_1 c_{a1}^2 + (1-p_1)c_{a2}^2 \quad . \tag{5}$$

Equations such as (5) are then used to obtain forms that express the departure SCV in terms of class 1. The paper offers alternative equations that involve different levels of complexity, as well as some that address different arrival models (e.g., bulk arrivals). These expressions use the arrival and service SCV for any number of classes at a given node to obtain the results. Once you have the departure SVC expressed by-class, one should be able to use that within QNA to obtain a congestion measure for each individual class. According to the paper, the purpose was to provide increased flexibility for QNA [51: 221].

So that we could choose the best departure approximation to incorporate into our version of QNA, a significant portion of our simulation model contains structures to measure arrival, service, and departure processes. However, from the simplest to most complex, the approximations do not match what we observe in the simulation. Table 3 shows a portion of the output from a run where the service rate of class 1 is $\mu_{1,c1} = 1$ and is $\mu_{1,c2} = 2$ for class 2. The other parameters are unchanged (Appendix E, Case 1).

Table 3: Queue 1 Arrival Process

Time Between Arrivals	Mean	Std Dev	SCV
For Both Classes	1.809	3.2155	3.1594
for Class 1	3.0156	5.7451	3.6296
for Class 2	4.5210	9.5232	4.4370
Proportion Class 1	0.599	0.0000	

Referring back to Equation (5), the left hand side should match the first SCV entry in the table when the last two entries are used on the right side with the probability of a class 1 customer being the proportion. The equation does not hold—the left does not equal the right. For this case, the only way to get a solution is to have $p_1 = 1.583$, which is nonsensical.

Even if we look at the equivalent information from the base case (Appendix C), where we did not differentiate class service times, we find again that the equation does not hold. There, the only way to have a solution to (5) for the arrival process at node 1 is for $p_1 = 2.59$. Since the proportion is an estimator of the probability that a customer is of class 1, we see that there is fundamental problem when using (5).

We turned to some of the more complicated approximations of [51] in hopes of finding a resolution to this dilemma. Unfortunately, we encounter the same outcome repeatedly—the approximations do not work in general. Additionally, we offer the following observation: If you have the tools available to obtain the information needed to use the approximations in [51], you do not need to be using this method because you already have the ability to obtain whatever performance estimates you desire.

What causes the departure approximations not to match? In a word, feedback. The SCV values obtained from the simulation contain the effects due to the feedback stream. By removing the feedback stream and having the network appear as serially connected, we get good agreement. Thus, there appears to be some modification required in order to use these departure approximations. Without a direct way to map the simulation SCVs under feedback into the framework of QNA, we cannot use the results of [51].

Including Customer Classes, II

However, the effort spent examining the SCV relationships is not a total loss. One thing we noticed relates to the probability, or proportion, of a customer class at a given node. In retrospect, it is common sense, and we get to why this is important after some discussion. If $\lambda_{i,c1}$ denotes the arrival rate of class 1 customers to node i, and $\lambda_{i,c2}$ is the rate for class 2, we find that there is an invariant relationship for the proportion. Namely,

$$P(class\_1 \mid \lambda_{i,c1}, \lambda_{i,c2}) = \frac{\lambda_{i,c1}}{\lambda_{i,c1}+\lambda_{i,c2}} \quad . \tag{6}$$

Yes, the values for the arrival rates for each class at a given node are known. If you refer to Appendix A, where we illustrate calculation of the internal arrival rates for the base case, you will see that those calculations hold independently for each class. To check this, obtain the λ values for each class in isolation and then add them. The result is the correct arrival rate for the particular node. If you desire to check this against simulation output, use the reciprocals of the mean values on the lines labeled as arrival processes.

To verify Equation (6), we ran 11 arrival combinations that caused the probability of being class 1 to vary from zero to one. Then, we varied the probability of departing the network from 0.30 to 0.70 for the 11 combinations. Finally, we checked for effects due to different service rates for each class. Recall that the simulation model is only structured for two classes. The worst single-run error noted was 0.8 percent.

The base case network is simple with respect to Equation (6) because we only admit customers at the leading node. However, the method generalizes to allow external arrivals at any node. Note that the procedure to obtain the internal traffic rates in Appendix A is equivalent to that of QNA in Appendix D, but QNA uses a matrix form. In a roundabout manner, because difficulty arises in keeping the classes identified in the matrix, we can inject different classes at different nodes without the other classes present to get individual rates at each node. Then when we include all classes together, we can still track the proportion of each class.

Next, we need to get to why this is important, but we first give some notes on authorship. Although developed independently here, we belatedly discovered that Glenenbe and Mitrani proposed this exact method in their procedure to admit multiple classes to their diffusion approximation [15: 144-148]. Ours is a simplification of theirs since they also employ the class-switching feature noted in BCMP [3]. Additionally, they deal with the problem of indicating which class enters which queue by multiplying the external arrival vector (see Appendix D) with a matrix that gives the probability that a particular class enters a given queue. However, this introduces complications because of how we built our representation of QNA so we have yet to incorporate the technique.

Hoping to take advantage of this finding, we looked at how Glenenbe and Mitrani dealt with obtaining congestion measure estimates. Their methodology follows what we envisioned, but one assumption used to get congestion measures by class is not true in general. In the text they state, "...a customer's waiting time at each queue does not depend on its class...[15: 149]," and from this observation they obtain estimates based on customer class. For heavy traffic, simulation shows this is a plausible assumption, but this is not true for moderate traffic as shown in Table 4 (Appendix E, Case 1a).

Table 4: Waiting Time by Class

Wait in Queue 1	Mean	Std Dev	Std Error
for Both Classes	0.681	0.024	0.007
for Class 1	0.718	0.032	0.010
for Class 2	0.625	0.012	0.004

There are only ten replications here, but a paired-t test on the class 1 and 2 wait says that we reject they are the same (p-value $\cong 0$). The only assumption here is normality.

Without even bothering to bring conditional expectation into the discussion, you probably could guess that the wait at a queue is the weighted average of each customer class' wait. For node i with customer classes of proportion $p_{i,cx}$, the expected wait in the queue is found to be

$$EW_i = p_{i,c1}EW_{i,c1} + p_{i,c2}EW_{i,c2} \quad . \tag{7}$$

If we can determine the waiting time of each class without simulation, we will be able to estimate all other congestion measures. For example, the number waiting at each node is found by multiplying (7) by the total arrival rate to the node, and the components of (7) become the length of the queue given by class.

Including Customer Classes, III

By knowing the probability, or proportion, of the classes at a queue, we know how the congestion measures are related to the waiting time of each class. We can get that level of detail from the simulation, but not from the approximation. Before addressing that, however, we must consider how to incorporate multiple classes into QNA under probabilistic routing.

What we want to do here is show that you can *aggregate* several classes into one generic customer class and still use QNA to provide congestion estimates. The user provides as input to QNA the means and the SCVs of all external arrival processes and all service processes. For this effort, we only use Poisson arrivals, which gives an SCV of one regardless of the number of classes contained in a stream. Therefore, we turn our attention to the service distributions.

QNA is structured to process only one class having probabilistic routing. Although, it allows the class to have any service time distribution, which is represented only by the mean and the SCV. Notice that a combination of customer classes having different service time distributions will have some generic mean service time and a corresponding SCV.

For each node i, the expected value of the service time, T, is given by

$$ET_i = p_{i,c1}ET_{i,c1} + p_{i,c2}ET_{i,c2} \quad . \tag{8}$$

This is just the conditional expectation given the average service time for each class. For our two-node base case, we used $\mu_i = 1 / ET_i$ as the input to QNA.

Continuing with conditional probability as the basis for combining the customer classes at a node, we have that the SCV, c^2, of service is calculated as

$$c_{s\,i}^2 = \frac{ET_i^2}{(ET_i)^2} - 1 \tag{9}$$

where

$$ET_i^2 = p_{i,c1}ET_{i,c1}^2 + p_{i,c2}ET_{i,c2}^2 \quad . \tag{10}$$

Again, this is just conditional expectation, but with a provincial method of notation. Here, you use the analytic expressions for the mean and variance of each class's service distribution in the identity $ET^2 = Var(T) + (ET)^2$ to form each component of the right hand side of the expression. For example, if the service times at node i for each class x are distributed exponentially with rate $\mu_{i,cx}$, then the expected service times and variance are $ET_{\,i,cx} = 1/\mu_{i,cx}$ and $Var(T) = 1/\mu^2_{\,i,cx}$, respectively. The terms on the right hand side become $ET^2_{\,i,cx} = 2/\mu^2_{\,i,cx}$. This technique is valid for any continuous theoretical distribution, but you only want to use non-negative, or properly truncated, distributions since these are service times.

As a check of the equations, we obtained the analytic result for (9) for the combination of a normal and an exponential distribution. We then used a portion of our simulation to generate observations from such an unlikely mix and checked against (9). For all practical purposes, the congruence was exact. We exclude that output from this work to preclude confusion due to the implied use of negative service times, but one can perform a similar check using Excel®.

Including Customer Classes, IV

Using Equations (8)-(10), we should be able to take any number of classes and express them as one class for use in QNA. In the previous section, we presented the method for finding the service time requirement at each node. However, QNA accepts external arrival SCVs in the same manner as service SCVs—as input. The same procedure applies to calculating the arrival SCVs if other than Poisson arrivals are used. For internal arrival, QNA calculates coefficients in the traffic variability equations (Appendix D) using the external arrival and the service SCVs.

Table 5 shows a portion of the output from a run where the service rate for class 1 is $\mu_{1,c1} = 1$ and is $\mu_{1,c2} = 2$ for class 2. The other parameters are unchanged from the base case (the output was mentioned earlier and is in Appendix E, Case 1).

Table 5: Queue 1 Service Process

Service Time	Mean	Std Dev	SCV
for Both Classes	0.8008	0.8731	1.1887
for Class 1	1.0006	1.0016	1.0019
for Class 2	0.5012	0.5014	1.0110
Proportion Class 1	0.5999	0.0000	

Although the rates are different for the two classes, both have service times from exponential distributions. Note that the SCV values are near one for each class, but the SCV for the combination is not.

Using Equations (8)-(10), we get that $\mu_1 = 1.25$ and that $c^2 = 1.1875$ for the service of two classes through node one of our two-node network. This is calculated assuming that the proportion of class 1 is exactly 0.60. These values differ by 0.10 percent of the simulation values.

Using the calculated values in QNA, we can compare our estimates from simulation and approximation. This is shown in Table 6 reusing the 10 replications discussed earlier (Appendix E, Case 1a).

Table 6: Different Service Rates at Node 1

Congestion Measures	Simulation		QNA
	10K Depart, 10 Reps		
	Mean	Std Error	Mean
EN_1	0.377	0.004	0.391
EW_1	0.681	0.007	0.7038
EN_2	0.020	0.000	0.0196
EW_2	0.052	0.001	0.0505
ER	5.810	0.032	5.9081

Notice that node 1, the node in feedback, has the largest discrepancy with respect to the simulation (approximately 3.7 percent). However, the expected response time only differs by 1.6 percent.

Modifying the previous example, let us now add different rates at node 2. The service rate of class 1 is $\mu_{2,c1} = 3$ and is $\mu_{2,c2} = 1$ for class 2. Using the same procedure, we get that $\mu_2 = 1.667$ and that $c^2 = 1.5926$. The results are shown in Table 7 for the output in Appendix E, Case 2 (note that we increased the number of departures).

Table 7: Different Service Rates at Both Nodes

Congestion Measures	Simulation		QNA
	50K Depart, 10 Reps		
	Mean	Std Error	Mean
EN_1	0.387	0.002	0.3935
EW_1	0.697	0.004	0.7084
EN_2	0.087	0.000	0.0935
EW_2	0.225	0.001	0.2404
ER	6.910	0.018	6.9888

Referring to Table 7, notice now that node 2, the downstream node of Figure 1, has the largest discrepancy with respect to the simulation (approximately 7.5 percent). Now, the expected response time differs by only 1.1 percent (a minor improvement with a higher number of departures).

Most of the discrepancies lie in the calculation of internal arrival SCVs. We discussed modifying those calculations in the section entitled Including Customer Classes, I. There, we noted that the methods of [51] showed no correspondence to the observed simulation output. Therefore, we elected to use the alternate method just illustrated. It is key to note that the errors observed in our multi-class QNA are consistent with the findings of the single-class product as given in [50].

Going Over the Main Points

For a multiple class queueing network having FCFS queues and class dependent service rates, an exact analytical solution does not exist. We found that QNA afforded us the greatest flexibility in a decomposition-approximation algorithm for inclusion of a multi-class methodology.

The creator of QNA proposed a method of including multiple class. However, we found that we could not incorporate the changes due to an inability to reconcile the method with observed simulation measures.

By observing relationships regarding the flow of customers through the network, Equation (6), we developed a procedure to admit multiple classes, Equations (8)-(10). This procedure reduces many classes into one class, with the proper parameters, to work with the existing QNA framework. The observed results are consistent with the original performance of QNA.

To verify our findings, we constructed a base case network known to be problematic to QNA, but known to have an exact solution with one customer class. In addition, we constructed a high-resolution simulation model and compared it against the base case over a wide range of system loading. Using the simulation as our indicator of the *true* state after including class-dependent service times, we then evaluated the performance of QNA using our procedure.

To date, however, we can only obtain estimates of the network congestion for the composite customer class. Referring to Equation (7), we noted that we need to identify the waiting times of each class in order to provide estimates on a by-class basis. This task is beyond the time limitations provided for this research.

Therefore, we offer our procedure as a tool to assist in developing a simulation study for a complex queueing network. The next section illustrates this methodology on a larger network.

Using the Procedure to Focus a Simulation Study

Discussion. In most simulation studies, you want to build a model having the least complexity necessary to accomplish the task. You may be interested in collecting detailed information on different customer classes, but you only want this at nodes meeting some criteria (e.g., longest queue). To do this, you could build a basic model and add class distinctions as you learn more about the how the network operates. However, this could become time consuming, particularly in verifying routing paths. What if you could quickly identify candidate locations before building the simulation model?

Whether existing or planned, you need a schematic representation of your network to build a simulation model or to apply any other analysis technique. Figure 3 shows the schematic of the network studied here.

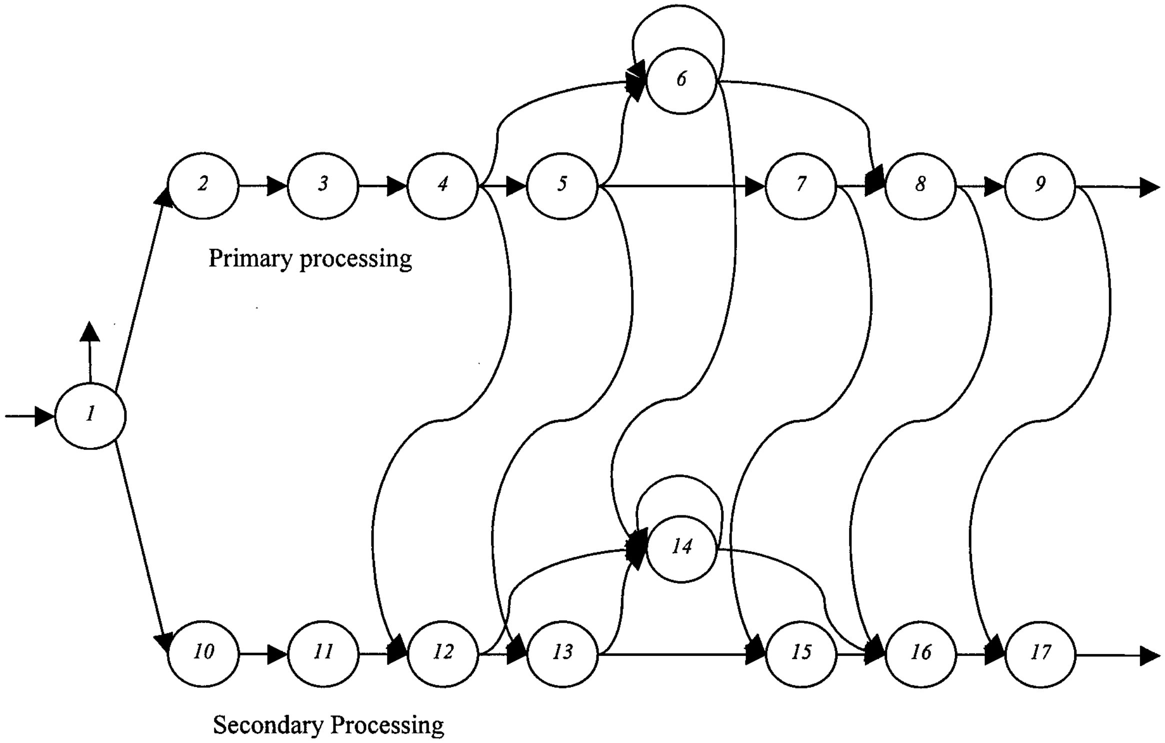


Figure 3: Evaluation Network

Referring to Figure 3, the numbered components represent single-server queues with FCFS discipline. They are the same as those in Figure 1, but without the explicit representation of the queue. Routing in the network is probabilistic, and arrows that do not end on a node indicate a departure point. All customers enter through node 1, and they may be from any number of classes. Notice that the upper and lower branches are identical in layout, but customers may transfer from the *primary* to the *secondary* legs. Notice, also, that nodes 6 and 14 have immediate feedback. However, there are currently no instances of delayed feedback.

For simulation, Appendix F (Basic Model) shows the minimum structure needed to begin a study of this network (node numbers shown in the queue symbol). You can measure average queue lengths, waiting times, and traffic intensities. Plus, the simulation model includes features to capture average response times and to count the number of customers passing through certain points. However, this model does not allow you to have class-dependent service times, nor does it allow you to capture any information peculiar to a class.

Say, for instance, it is important to know how many customers of a class x are contained in the average queue length of node i. This is given exactly by

$$EN_{i,cx} = \lambda_{i,cx} EW_{i,cx} \quad . \tag{11}$$

This is a direct consequence of Equation (7), and is known as Little's Law [17]. However, you can only apply this equation if you include the necessary structure in the simulation model. That is what the proposed method is about—providing a tool that helps identify where to add complex, multi-class structure to the simulation model.

Development. In order to use our decomposition approximation, we must have an explicit representation of the routing matrix (Table 8). Recalling our reconstruction of QNA, as shown in Appendix D, one could replace that routing matrix with a larger one. However, it is just as simple to read a flat file generated from a spreadsheet, which simplifies adding or deleting nodes.

Table 8: Routing Matrix

node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1		0.3								0.4							
2			1														
3				1													
4					0.3	0.3						0.4					
5						0.3	0.3						0.4				
6						0.4		0.3						0.3			
7								0.5							0.5		
8									0.5							0.5	
9																	0.5
10											1						
11												1					
12													0.5	0.5			
13														0.5	0.5		
14														0.5		0.5	
15																1	
16																	1
17																	

The matrix will always be *nxn* (17x17 here), which is one reason to number the nodes. Notice that you do not include the probability of departure to the outside world; thus, rows may sum to less than one. Contrary to a simulation model, you can quickly add and remove paths, as well as nodes (change matrix, save, check results). This alone **allows rapid exploration of scenarios**.

For external arrivals, let us say there are 85 customer classes arriving to node 1. The first class, class 1, has an arrival rate of $\lambda_{c1} = 5$. For the sake of sanity, we assume that the other 84 classes are similar enough to be identical. Their aggregate arrival rate is $\lambda_{c2} = 8.4$ (each has an individual rate of 0.10). This gives the total external arrival rate as $\lambda_o = 13.4$. All arrival processes are Poisson.

The service time distributions are exponential. Node 1 has a high service rate since it feeds both legs, $\mu_1 = 50$, and is equal for both classes. The rates at other nodes are class-dependent, but for now let them be 20 for both classes. However, say the service rates are actually $\mu_{14,c1} = 10$ and $\mu_{14,c2} = 20$ for the two classes at node 14. Do we need to augment the simulation model at this node?

This service process has a weighted average rate of $\mu_{14} = 14.56$ and a SCV of $c^2 = 1.2481$. Here, we have used Equation (6) to calculate the probability of being class 1 as 0.373 (we only have input at node 1 so this holds for all nodes in this case), and then we used Equations (8)-(10) to obtain the aggregate mean service time and SCV.

On the same spreadsheet that contains the routing matrix, we create four column vectors to contain the external arrival and service information. Table 9, on the next page, illustrates what we need for this example. Here, we also included the node names. Again, if you are using a product like Mathcad®, you could change our sample (Appendix D) to read the vectors from a file.

Table 10 gives the output from QNA using the routing of Table 8 and the columns from 9. Referring back to the routing matrix, note that 30 percent of all arrivals exit the system at node 1 and 30 percent go to the primary processing leg. However, notice that the expected number at node 14 (secondary leg) is 32 times larger than at any of the preceding nodes. Clearly, we need to add detail to the simulation model at this node (we could check other nodes also).

Table 9: Arrival and Service Information

Node	Node Name	External Arrivals	Arrival SCV	Service Rates	Service SCV
1	SC0	13.4	1	50	1
2	SC1A	0	1	20	1
3	SC2A	0	1	20	1
4	SC3A	0	1	20	1
5	SC4A	0	1	20	1
6	SC5A	0	1	20	1
7	SC6A	0	1	20	1
8	SC7A	0	1	20	1
9	SC8A	0	1	20	1
10	SC1B	0	1	20	1
11	SC2B	0	1	20	1
12	SC3B	0	1	20	1
13	SC4B	0	1	20	1
14	SC5B	0	1	14.56	1.2481
15	SC6B	0	1	20	1
16	SC7B	0	1	20	1
17	SC8B	0	1	20	1

Table 10: QNA Congestion Estimates

Node	Node Name	EN (QNA)	EW (QNA)
1	SC0	0.098	0.007
2	SC1A	0.051	0.013
3	SC2A	0.051	0.013
4	SC3A	0.051	0.013
5	SC4A	0.004	0.003
6	SC5A	0.020	0.008
7	SC6A	0.000	0.001
8	SC7A	0.002	0.003
9	SC8A	0.001	0.001
10	SC1B	0.098	0.018
11	SC2B	0.098	0.018
12	SC3B	0.186	0.027
13	SC4B	0.049	0.012
14	SC5B	5.957	0.477
15	SC6B	0.013	0.006
16	SC7B	0.363	0.041
17	SC8B	0.389	0.043
		Expect Response Time	
		0.861	

Comparison to a Single Simulation Run. Assume that we are satisfied that node 14 is the problem node, and now we want to build the simulation model. Appendix F contains a section at the end called Modifications. This section shows the changes needed convert the basic simulation model to capture useful information at node 14 by-class. What is not shown, however, are the other node pointer changes that you must perform in order to ensure all routes now reference the node that assigns the class-dependent service times for node 14. We do not want to add or move structures like this ad hoc because there are many opportunities for error, which is why the proposed method is beneficial. Table 11 compares our previous finding to the altered simulation model (Appendix G, Case 1).

Table 11: QNA Congestion Estimates Compared to Simulation

Node	Node Name	EN (QNA)	EW (QNA)	EN (Sim)	EW (Sim)
1	SC0	0.098	0.007	0.099	0.007
2	SC1A	0.051	0.013	0.049	0.012
3	SC2A	0.051	0.013	0.050	0.013
4	SC3A	0.051	0.013	0.050	0.012
5	SC4A	0.004	0.003	0.004	0.003
6	SC5A	0.020	0.008	0.018	0.007
7	SC6A	0.000	0.001	0.000	0.001
8	SC7A	0.002	0.003	0.002	0.002
9	SC8A	0.001	0.001	0.001	0.001
10	SC1B	0.098	0.018	0.099	0.019
11	SC2B	0.098	0.018	0.099	0.019
12	SC3B	0.186	0.027	0.188	0.027
13	SC4B	0.049	0.012	0.049	0.012
14	SC5B	5.957	0.477	5.899	0.476
15	SC6B	0.013	0.006	0.013	0.006
16	SC7B	0.363	0.041	0.379	0.043
17	SC8B	0.389	0.043	0.389	0.043
		Expect Response Time		Expect Response Time	
		0.861		0.858	

Note that the estimate of the expected number at node 14 only differs from the simulation by 0.98 percent, although node 14 is now in heavy traffic ($\rho_{14} = 0.8584$ from QNA and 0.854 from simulation) and has immediate feedback. The worst estimate is at node 6 (11 percent), which also is in immediate feedback. Note that these comparisons are based on a single, long simulation run. We will discuss replications shortly.

What we are seeing at these nodes are effects due to variability as observed in the simulation. At node 6, QNA still views the node as having an exact solution because we did not include class-dependencies at, or prior, to the node (SCV of arrival process is one). However, the simulation captures some variability in the arrival process due to the structure of the network. The effect is moderated at node 14 because we explicitly included an SCV differing from one for the service process. This then factors into the arrival SCV calculation at node 14.

Since we modified our simulation model to capture class-specific information at node 14, let us present part of that in Table 12 (Appendix G, Case 1).

Table 12: Node 14 Expected Wait

Expected Wait Time	Mean
for Both Classes	0.476
for Class 1	0.491
for Class 2	0.467
Proportion Class 1	0.375

In Table 12, we see that the proportion of class one customers is actually 0.375 at node 14. This is less than 1 percent from our estimate of 0.373, and is an effect of the short-term variability the model is seeing. The observed simulation arrival rate to node 14 is 12.38 instead of the long run value of 12.50—number of arrivals is slightly low.

For class 1, we have that the observed arrival rate is 4.65 (theoretical value is 5). Using this observed value in Equation (11) along with the expected wait for class 1, we find that there are 2.28 class 1 customers waiting at node 14 on average. The balance of the 5.899 belong to class 2 (approximately 61 percent). Interestingly, the percentages for the expected number of each class always seem to lie close to the proportion of each class. However, the two quantities differ by some parameter not yet identified.

How accurate was our procedure to combine the service rates of the classes, Equations (8)-(10)? Recall that we used a service SCV of $c^2 = 1.248$ and a mean rate of $\mu_{14} = 14.56$ in QNA, and we modeled this in the simulation by explicitly assigning different rates to the two classes. Table 13 shows that the mean (1/14.56 = 0.0687) and SCV match extremely well. Note that the SCV must be checked using the standard deviation of a run and not a replication summary output.

Table 13: Node 14 Service Times

Service Time	Mean	Std Dev	SCV
for Both Classes	0.069	0.077	1.245
for Class 1	0.100	0.101	1.020
for Class 2	0.050	.050	1.000

Comparison to a Replication. To compare the estimates to the QNA values, we need to have a longer run to alleviate the arrival variability. We still could not obtain confidence intervals using batch-means on a single run because properly parsing a long run remains problematic. To obtain a statement of confidence, we need replication. In our previous study of the two-node network, we used the technique of common random numbers. This was necessary to compare changes on an equal footing. In other words, we fixed the starting point of the random number streams so that we only observed changes due to parameters and not changes due to different random number draws. However, that limited our ability to replicate long simulation runs. For this network, we did not use that technique so replication is appropriate. Table 14 shows this information.

Here, we have increased the number of departures to 250,000, and we replicate the run 10 times (Appendix G, Case 2). The left-hand columns are from Table 10.

Table 14: Congestion Estimates under Replication

Node	Node Name	EN (QNA)	EW (QNA)	EN (Replication)	Std Err (Replication)	EW (Replication)
1	SC0	0.098	0.007	0.099	0	0.007
2	SC1A	0.051	0.013	0.051	0	0.013
3	SC2A	0.051	0.013	0.051	0	0.013
4	SC3A	0.051	0.013	0.051	0	0.013
5	SC4A	0.004	0.003	0.004	0	0.003
6	SC5A	0.020	0.008	0.020	0	0.007
7	SC6A	0.000	0.001	0.000	0	0.001
8	SC7A	0.002	0.003	0.002	0	0.003
9	SC8A	0.001	0.001	0.001	0	0.001
10	SC1B	0.098	0.018	0.098	0.001	0.018
11	SC2B	0.098	0.018	0.098	0	0.018
12	SC3B	0.186	0.027	0.187	0.001	0.027
13	SC4B	0.049	0.012	0.049	0	0.012
14	SC5B	5.957	0.477	5.808	0.079	0.465
15	SC6B	0.013	0.006	0.013	0	0.006
16	SC7B	0.363	0.041	0.375	0.003	0.042
17	SC8B	0.389	0.043	0.400	0.002	0.044
		Expect Response Time		Expect Response Time		
		0.861		0.852		

Note that most values now compare identically to QNA in the mean. We see that node 14 still does not agree, but recall that it is in heavy traffic. However, note that a 95 percent confidence interval at node 14 still includes the QNA value. This indicates that we may need to try starting the simulation with some customers present to correct this bias. The same is not true for nodes 16 and 17; the 95 percent intervals do not capture the

QNA estimate. To correct this, we need to further develop the departure SCV approximations to account for classes. This relates to the work of Whitt in [51].

For the expected waiting time at node 14, we have the replication values overall and by class in Table 15. The aggregate waiting time is less than the QNA value, but we see that a 95 percent confidence interval captures the QNA estimate. The class-dependent estimates are not independent from each other; therefore, we use a paired-t test to assess whether the class 1 wait statistically differs from the class 2 wait. Using the data from the individual runs (this portion is included in the appendix), we find that the waiting times of the two classes are different (p-value $\cong 0$).

Table 15: Node 14 Expected Wait

Expected Wait Time	Mean	Std Err
for Both Classes	0.465	0.006
for Class 1	0.477	0.006
for Class 2	0.457	0.006
Proportion Class 1	0.373	0.001

It is interesting to note that class 1 has a longer wait although class 2 customers are processed twice as fast. Checking the expected number waiting at node 14, Equation (11) shows that there are 3.59 class 2 customers and 2.21 class 1 customers. Using information such as this, we have evidence indicating that the class with the larger arrival rate has a shorter average wait because the other class finds that it must wait at a queue already occupied (or, some similar comparison). Regarding service, the SCV under replication is 1.246 with a standard error of 0.002 (not part of the summary output). This compares extremely well with our input SCV of 1.248.

Adding Delayed Feedback. What happens if we include delayed feedback? What if we change the routing matrix to have 25 percent (or more appropriately, a probability of 0.25) of all customers completing node 9 return to node 1 and start the process again? Table 16 gives the QNA response compared to the case shown in Table 10.

Table 16: Congestion Estimates under Delayed Feedback

Node	Node Name	EN	EW	EN (Table 10)	EW (Table 10)
1	SC0	0.100	0.007	0.098	0.007
2	SC1A	0.052	0.013	0.051	0.013
3	SC2A	0.052	0.013	0.051	0.013
4	SC3A	0.052	0.013	0.051	0.013
5	SC4A	0.004	0.003	0.004	0.003
6	SC5A	0.020	0.008	0.020	0.008
7	SC6A	0.000	0.001	0.000	0.001
8	SC7A	0.002	0.003	0.002	0.003
9	SC8A	0.001	0.001	0.001	0.001
10	SC1B	0.100	0.019	0.098	0.018
11	SC2B	0.100	0.019	0.098	0.018
12	SC3B	0.191	0.027	0.186	0.027
13	SC4B	0.050	0.013	0.049	0.012
14	SC5B	6.423	0.509	5.957	0.477
15	SC6B	0.013	0.006	0.013	0.006
16	SC7B	0.372	0.041	0.363	0.041
17	SC8B	0.400	0.043	0.389	0.043
		Expect Response Time		Expect Response Time	
		0.909		0.861	

Note that the difference between the two scenarios is negligible. In fact they are similar enough that we would not bother changing the simulation to check this unless it was an essential piece of information.

Many of the discrepancies we have seen relate to transient behavior remaining in the simulation, without replication. The proposed modification to QNA admits multiple customer classes, and can serve as a useful tool to focus a simulation study.

Methodology Flowchart

Figure 4, originally from Law and Kelton, illustrates the choices generally available in the analysis of any system [29: 4]. We have modified the illustration to properly include approximation techniques as a choice. Additionally, we show approximation as an alternate path to reach simulation from the mathematical model formulation, which represents our methodology as applied to queueing networks. Figure 5 then shows the process representation along this path to employ our methodology.

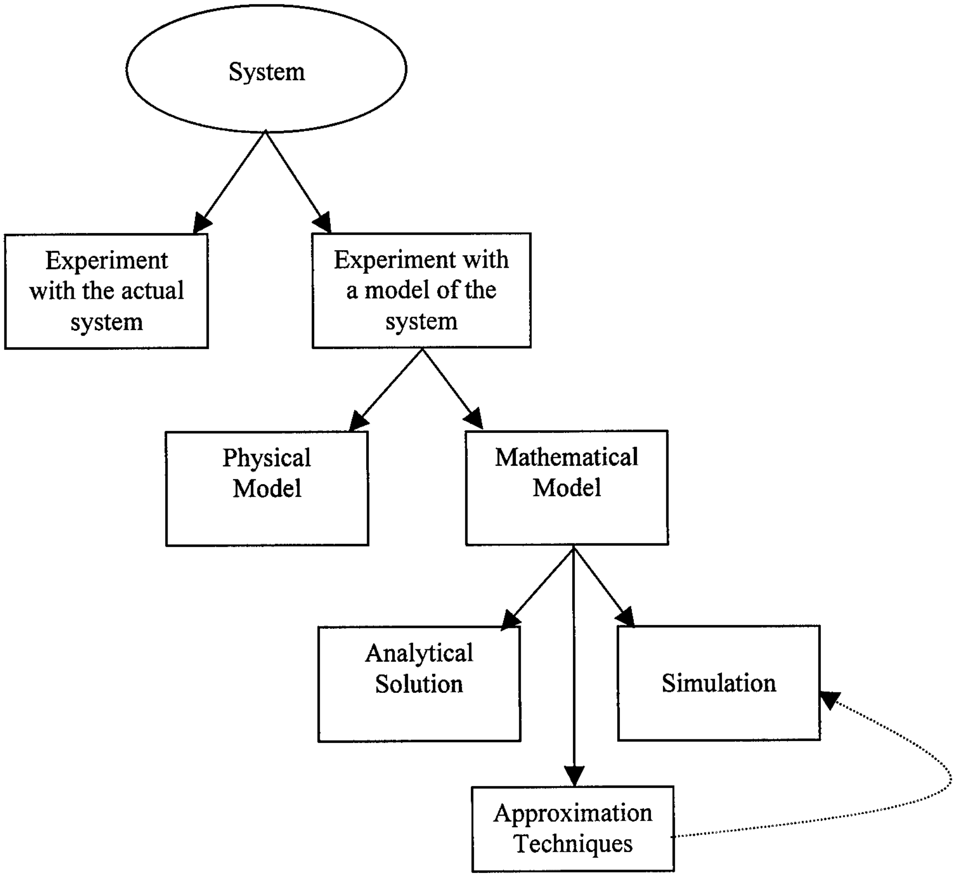


Figure 4: Ways to Study a System

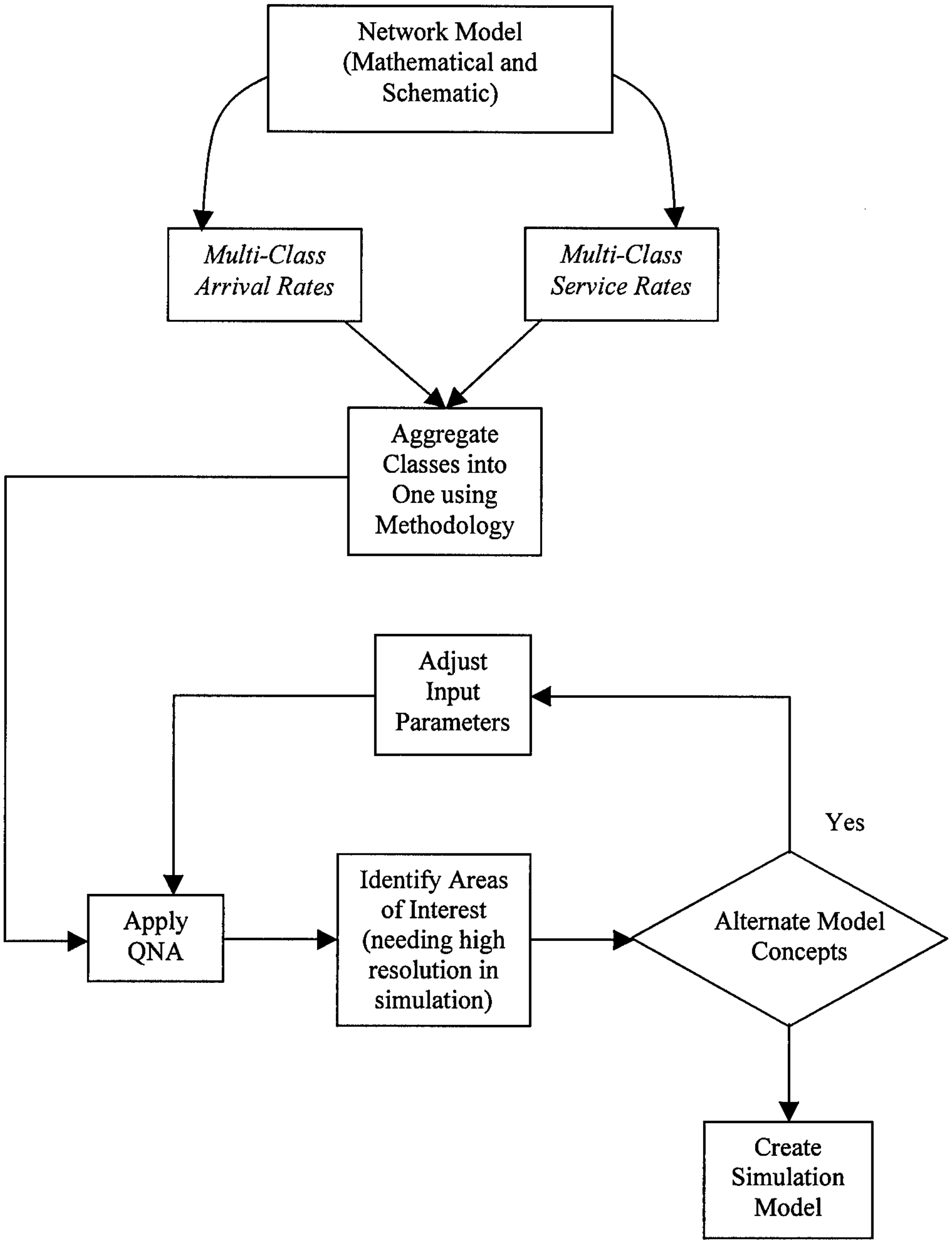

Network Model (Mathematical and Schematic)
Multi-Class Arrival Rates
Multi-Class Service Rates
Aggregate Classes into One using Methodology
Adjust Input Parameters
Yes
Apply QNA
Identify Areas of Interest (needing high resolution in simulation)
Alternate Model Concepts
Create Simulation Model

Figure 5: Flowchart of Methodology Process

IV. Conclusion

Summary

In the study of complex queueing networks, one generally seeks to employ exact analytic solutions to reduce the burden on computational resources. Barring the existence of an exact solution, the alternatives include approximation techniques and simulation. Approximation is the more attractive alternative from a time and effort perspective; however, cases exist that are not amiable to this technique.

The objective of this thesis was to increase the flexibility of approximation techniques in obtaining estimates of congestion measures for complex, open queueing networks. The focus being on networks having several customer classes and class-dependent service times, but not having an exact solution. The associated goal was to provide a tool to easily obtain these estimates before building an intricate simulation model.

In most simulation studies, you want to build a model having the least complexity necessary to accomplish the analysis task. You may be interested in collecting detailed information on different customer classes, but you only want this at nodes meeting some criteria (e.g., longest queue). To do this, you could build a basic model and add class distinctions as you learn more about how the network operates. However, this could become time consuming, particularly in verifying routing paths.

The outcome of this effort is a methodology that allows you to quickly identify candidate locations requiring higher resolution results before building the simulation model. We accomplished this by developing a procedure that aggregates multiple classes into a single class in order to apply an existing approximation technique. The result is a tool that one can use to focus a simulation study. We found that the methodology results in good agreement with simulation models that explicitly represent multiple customer classes. The result is a tool that one can use to focus a simulation study. It provides easy evaluation of a particular network configuration, as well as rapid evaluation of alternative configurations.

Research Overview and Findings

The network models utilized in this effort were classic *Jacksonian* networks. They contained a finite number of unlimited capacity, single-server queue nodes connected in an arbitrary fashion. Routing between the queues was based on specified branching probabilities. The networks were open: customers enter from the outside world and all customers eventually depart. The service discipline at all queue nodes was first-come-first-serve (FCFS); however, the different customer classes could have different service distributions at any given node.

A review of relevant literature did not identify an exact analytic solution for the case described. The use of FCFS queues with class-dependent service times prevented obtaining an exact solution. Specifically, networks constructed as given do not have a particular type of solution known as the product-form. However, we also found that product-form and exact solution are synonymous except in rare circumstances. Here, in rare circumstances, small networks may be solved using their global balance equations. For this, the practical limit is on the order of a two-node network, but even that size may become too complicated in some cases.

Satisfied that an exact solution was unattainable, we reviewed various approximation techniques. For networks having service centers that are not continuously busy, decomposition algorithms provided reasonable estimates of network performance and afforded the greatest flexibility for modification. In particular, the algorithm for the Queueing Network Analyzer (QNA, [49]) was found suitable enough to serve as the basic structure for this study. For many networks, QNA could provide congestion measure estimates with errors not exceeding 10 percent relative to simulation values [50].

QNA allows for multiple customer classes with different service time distributions; however, the user must specify the exact route of each class. For general routing, such as used here, QNA only permits one customer class. Therefore, the task became one of combining general routing and multiple classes.

The author of QNA provided one concept of how to accomplish this combining task, and suggested equations that could be incorporated into the original QNA [51]. However, we found that we could not properly match simulation observations to the proposed equations. We observed the problem to be related to the effect of customers re-circulating in a network (feedback). The QNA algorithm appears to transform a network into a series of serially connected queues, whereas the simulation maintains a representation closer to the actual layout. The result is that the arrival, service, and departure measures found through simulation and calculated by QNA do not have an obvious mapping to each other.

From simulation results used to evaluate the paper mentioned in the previous paragraph, we noted some basic relationships in the data appeared useful. We observed that the probability of a customer class at a given queue followed a simple form. From conditional probability, we then observed that we could form the mean and variance for the aggregate service time of several customer classes having different individual service distributions.

We then hypothesized that this information could be used in the QNA framework to represent multiple customer classes as a single composite class. One major assumption of QNA is that the mean and squared coefficient of variation (SCV) contain adequate information to represent arrival, service, and departure processes. Where SCV is variance divided by the square of the mean. The validity of this assumption was proven prior to the development of QNA Therefore, the specific structure of QNA seeks to enhance estimates using the assumption. The accuracy of QNA is documented in [50].

To evaluate whether the distribution of our composite customer class in QNA represented reality as viewed by the simulation, we constructed two networks having explicit representation of customer classes. At worst, we found that our multi-class QNA was within the performance documented in [50]. At best, we noted that the multi-class QNA differed less than 1 percent from the simulation. This was an evaluation, not a formal test of the hypothesis.

In the instances where we did not get an exact match between the simulation and the multi-class QNA, we found that QNA faithfully represented the performance trend and correct magnitude. For example, say that simulation yields an average of 14.65 customers waiting in a given queue; the multi-class QNA may report 15.03.

Inconsistencies in estimation generally fall into two categories: heavy traffic problems and departure SCV estimation. In any situation where a queue experiences heavy traffic, the simulation and QNA estimates will not likely agree. This was caused more by the initial conditions and the length of the simulation run than with the formulations contained in QNA. However, problems also occurred under moderate traffic if the simulation run is too short to eliminate the startup transient.

We did note that departure SCVs calculated using our service SCVs (a procedure of the QNA algorithm) introduced actual bias into our results. This becomes particularly true when the network has feedback streams. Although the bias is within the bounds noted, this is a problem area requiring modification in a manner suggested by [51].

As a tool to focus a simulation study of a complex network, the method performs well. For large networks, reasonable performance estimates can be obtained quickly. Once the basic input parameters are determined, different scenarios may be rapidly evaluated in a fraction of the time needed to modify a typical simulation model. This allows one to check ideas and determine where to invest time and funding when constructing a simulation model to obtain performance estimates on a by-class basis.

Limitations

The methodology to admit multiple customer classes under probabilistic routing performs well in the QNA framework. Any number of classes may be combined into one composite class before applying the QNA algorithm. Moreover, the results are comparable (sometimes identical) to simulation estimates of congestion measures. However, we have not been able to later decompose the composite customer to give performance measures on a by-class basis. An apparent relationship to the probability of the class of a customer arriving at a queue was noted; however, a working theory did not evolve.

The method to aggregate multiple customer classes may fail if the service rate of one class is significantly higher than other classes. This becomes apparent in calculating the composite service SCV when a negative value is returned. In simple terms, the class having a lower rate dominates the service distribution at that queue—the queue does not seem to exist for the higher rate class. Thus, we develop an apparent inconsistency best resolved by simulation. QNA cannot handle zero-time service centers in general. However, this does not preclude using composite rates that are large. We sought to avoid this situation since it provided no useful insight other than as a limitation.

QNA only estimates long run, or steady state, network performance. Therefore, it is most useful for networks that operate for extended periods or those not having a notable startup transition phase. Additionally, we did not include the estimates of variability provided within the QNA formulation. Given the stated focus, that was deemed unnecessary.

QNA does not permit queueing disciplines other than FCFS. However, due to the abundance of work on product-form solutions, most other disciplines of practical interest have an exact solution. Additionally, QNA is currently restricted to infinite capacity queues.

As part of the scope of this effort, we restricted the study only to Poisson arrivals. Thus, the effectiveness of the class aggregation technique was not verified for general arrival distributions. QNA was developed to permit arrival distributions other than Poisson [1; 48; 49; 50], and is based on work extending back at least to Kelly [25]. Given that arrival SCVs combine in the same manner as service SCVs, the assumption that the procedure works for general arrival and service distributions is not unfounded.

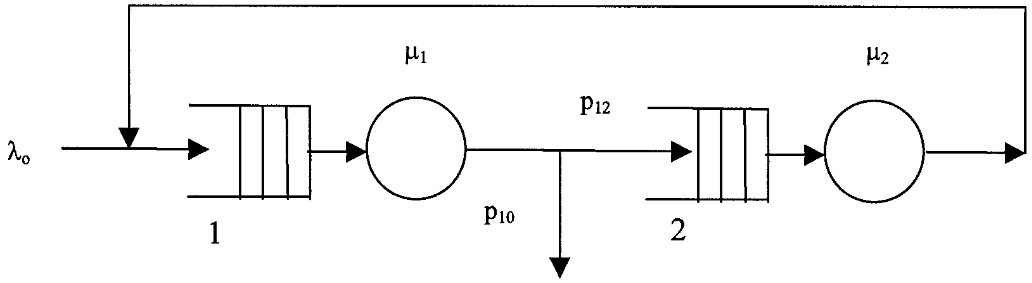
Recommendations

Obtaining greater accuracy for the congestion measure estimates appears to be possible by improving the departure approximations for the queues. This is the subject of [51]; however, simulation observations are not consistent with that methodology. Understanding that methodology and incorporating it within the QNA framework is the next logical step of this work.

Parallel to improving the methodology of including multiple classes is development of the methodology to obtain congestion measure estimates by class. We observed that there appears to be a connection to the probability of a particular class being at a given queue. However, we could not identify the actual relationship. If one could solve this problem, simulation studies to obtain these measures would not be necessary. That is, assuming one is content with approximations of long run performance to guide the development of a simulation study.

Originally, we desired to incorporate capacitated queueing (blocking with loss) simultaneously with class-dependent service time distributions at FCFS queues. However, it quickly became clear that the task of incorporating class dependencies overshadowed that feature. Thus, one direction of future research would be to incorporate this capability. Although not included in the literature review, we looked at several works in the area of capacitated queueing networks. However, most involved other forms of blocking mechanisms (e.g., repetitive service for a blocked customer). We did uncover some literature that a future researcher might begin with [4; 35; 36; 44].

Appendix A: Exact Solution for Base Case



Let λ be the superposition of two classes having Poisson arrivals with means interarrival times of 15 and 10. For each class, let both service distributions be exponential with rates of 2 and 3, respectively, with FCFS disciplines.

Then,

$\lambda := \frac{1}{15} + \frac{1}{10}$, or $\lambda = 0.1667$, with $\mu_1 := 2$ and $\mu_2 := 3$

Also, let the probability that a customer departs after completion at the first server, p_{10}, be 0.30.

As defined, this network has a product-form solution. Specifically,

$$p(n_1, n_2) := (1 - \rho_1) \cdot (\rho_1)^{n_1} \cdot (1 - \rho_2) \cdot (\rho_2)^{n_2}$$

Where the n_i are the number of customers at each node, and the ρ_i are the traffic intensities at each node.

However, we can use the simple results for two queues in tandem to find the expected queue lengths and system response time at steady state. This is accomplished by noting that the arrival rate to each queue is found by simultaneous solution of two simple equations. From there, the traffic intensity at each queue is found, which eventually leads to the desired measures [45: 412-417]. The solution are as follows:

$$\lambda_1 = \lambda + \lambda_2 \qquad \text{and} \qquad \lambda_2 = \lambda_1 \cdot p_{12}$$

Thus,

$$\lambda_1 = \frac{\lambda}{1 - p_{12}} = \frac{\lambda}{p_{10}} \qquad \text{and} \qquad \lambda_2 = \frac{p_{12} \cdot \lambda}{p_{10}}$$

where

$$p_{10} := 0.30 \qquad \text{and} \qquad p_{12} := 1 - p_{10}$$

Since the traffic intensity, ρ, is defined as the ratio of the arrival to the service rate, we have

$$\rho_1 := \frac{\lambda}{p_{10} \cdot \mu_1} \qquad \text{or} \qquad \rho_1 = 0.2778$$

and

$$\rho_2 := \frac{p_{12} \cdot \lambda}{p_{10} \cdot \mu_2} \qquad \text{or} \qquad \rho_2 = 0.1296$$

From these, we have that the expect number in each queue is

$$EN_1 := \frac{(\rho_1)^2}{1 - \rho_1} \qquad \text{or} \qquad EN_1 = 0.1068$$

and

$$EN_2 := \frac{(\rho_2)^2}{1 - \rho_2} \qquad \text{or} \qquad EN_2 = 0.0193$$

The expected waiting time at each queue is

$$EW_1 := \left(\frac{1}{\mu_1}\right)\cdot\left(EN_1 + \rho_1\right) \qquad \text{or} \qquad EW_1 = 0.1923$$

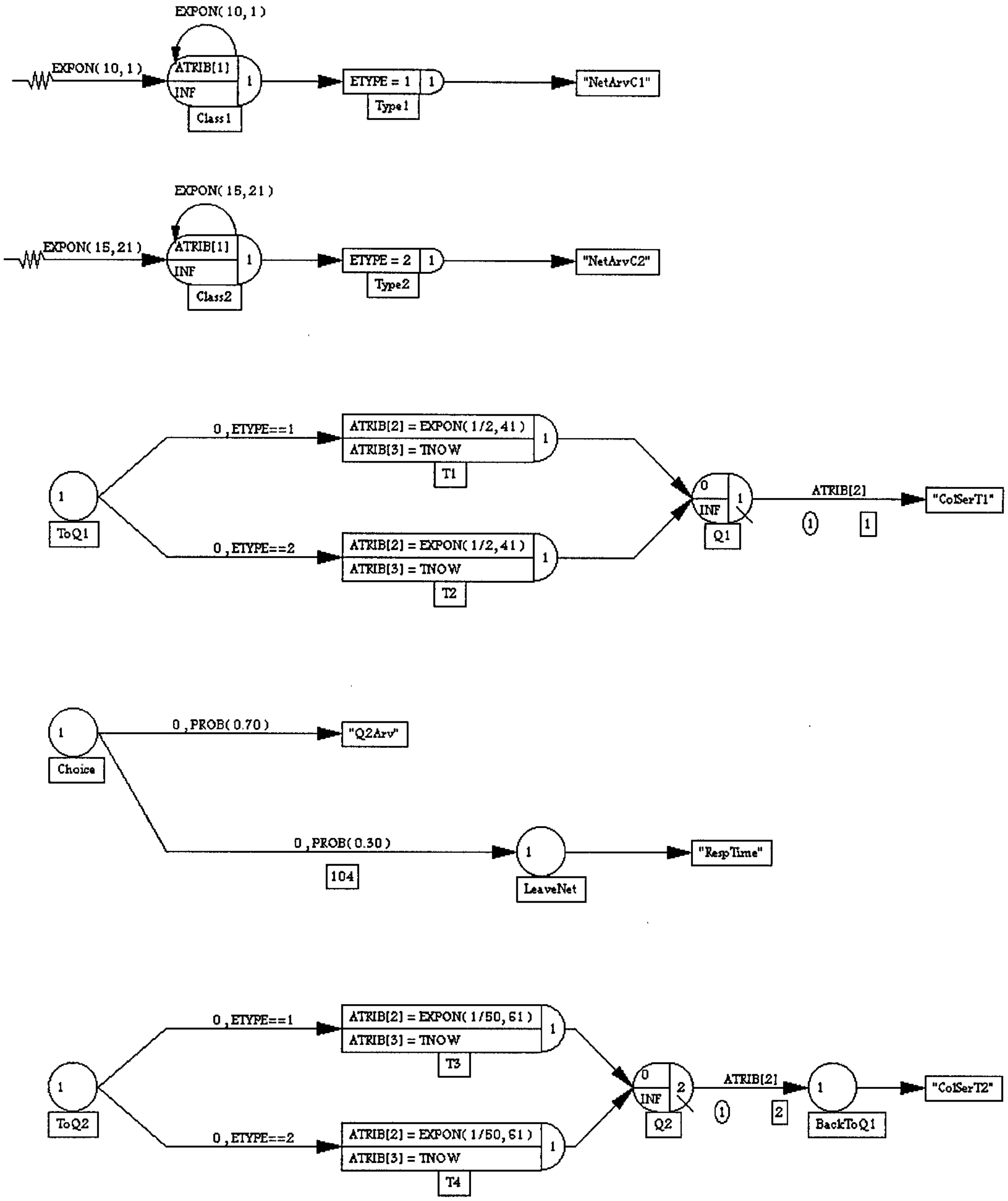
and

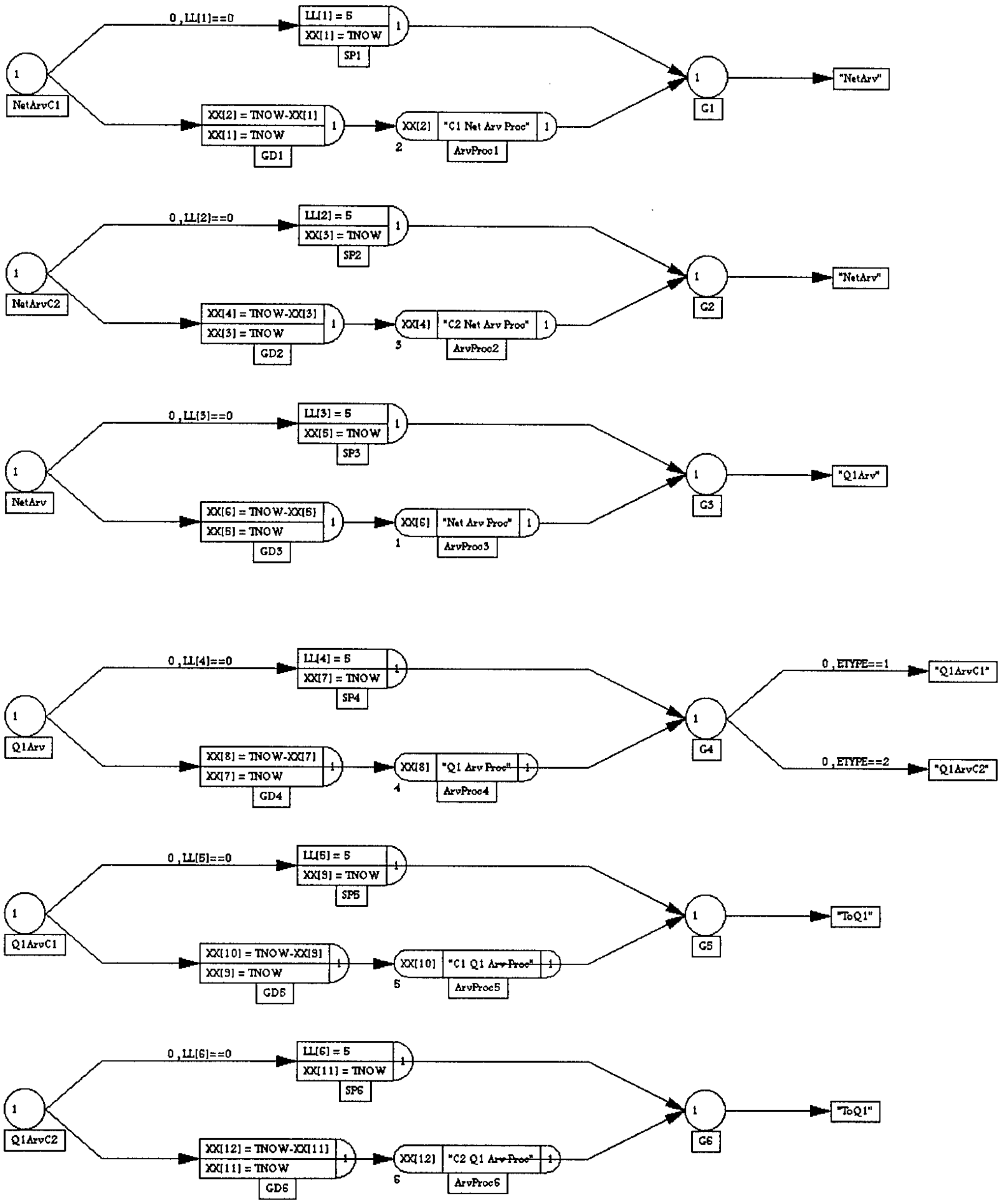
$$EW_2 := \left(\frac{1}{\mu_2}\right)\cdot\left(EN_2 + \rho_2\right) \qquad \text{or} \qquad EW_2 = 0.0496$$

Finally, the expected system response time is given by

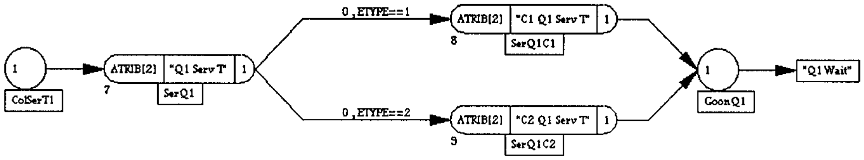
$$ER := \left(\frac{\rho_1}{1-\rho_1} + \frac{\rho_2}{1-\rho_2}\right)\cdot\frac{1}{\lambda} \qquad \text{or} \qquad ER = 3.2013$$

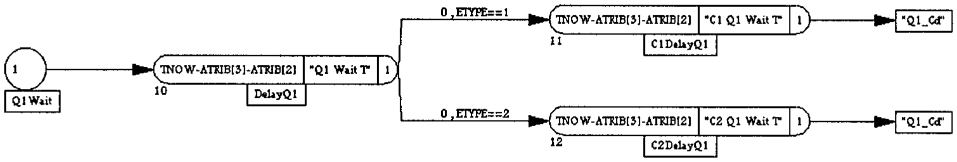
Appendix B: Two-Node Network Simulation Model

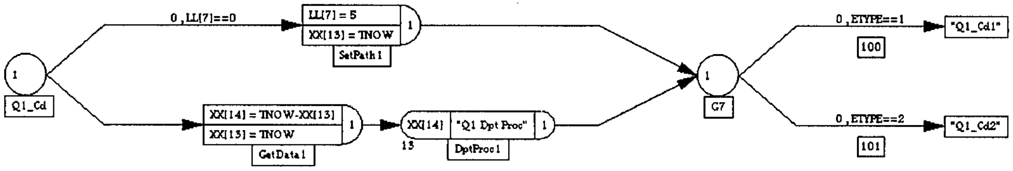


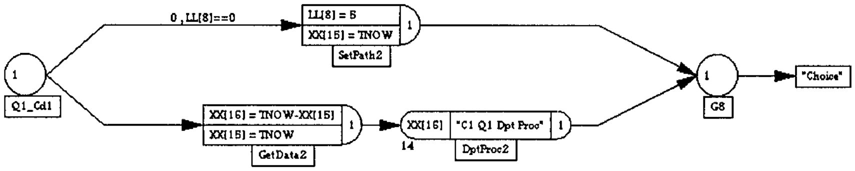


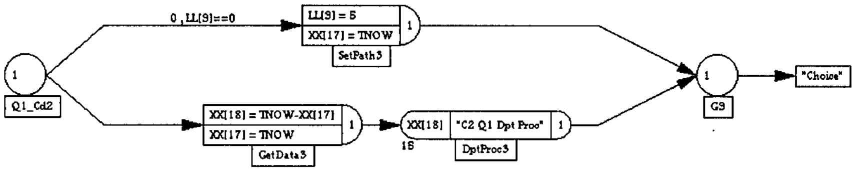
1
NetArvC1
0 ,LL[1]==0
LL[1] = 5
XX[1] = TNOW
1
SP1
XX[2] = TNOW-XX[1]
XX[1] = TNOW
1
GD1
XX[2]
"C1 Net Arv Proc"
1
2
ArvProc1
1
G1
"NetArv"
1
NetArvC2
0 ,LL[2]==0
LL[2] = 5
XX[3] = TNOW
1
SP2
XX[4] = TNOW-XX[3]
XX[3] = TNOW
1
GD2
XX[4]
"C2 Net Arv Proc"
1
3
ArvProc2
1
G2
"NetArv"
1
NetArv
0 ,LL[3]==0
LL[3] = 5
XX[5] = TNOW
1
SP3
XX[6] = TNOW-XX[5]
XX[5] = TNOW
1
GD3
XX[6]
"Net Arv Proc"
1
1
ArvProc3
1
G3
"Q1Arv"
1
Q1Arv
0 ,LL[4]==0
LL[4] = 5
XX[7] = TNOW
1
SP4
XX[8] = TNOW-XX[7]
XX[7] = TNOW
1
GD4
XX[8]
"Q1 Arv Proc"
1
4
ArvProc4
1
G4
0 ,ETYPE==1
"Q1ArvC1"
0 ,ETYPE==2
"Q1ArvC2"
1
Q1ArvC1
0 ,LL[5]==0
LL[5] = 5
XX[9] = TNOW
1
SP5
XX[10] = TNOW-XX[9]
XX[9] = TNOW
1
GD5
XX[10]
"C1 Q1 Arv Proc"
1
5
ArvProc5
1
G5
"ToQ1"
1
Q1ArvC2
0 ,LL[6]==0
LL[6] = 5
XX[11] = TNOW
1
SP6
XX[12] = TNOW-XX[11]
XX[11] = TNOW
1
GD6
XX[12]
"C2 Q1 Arv Proc"
1
6
ArvProc6
1
G6
"ToQ1"

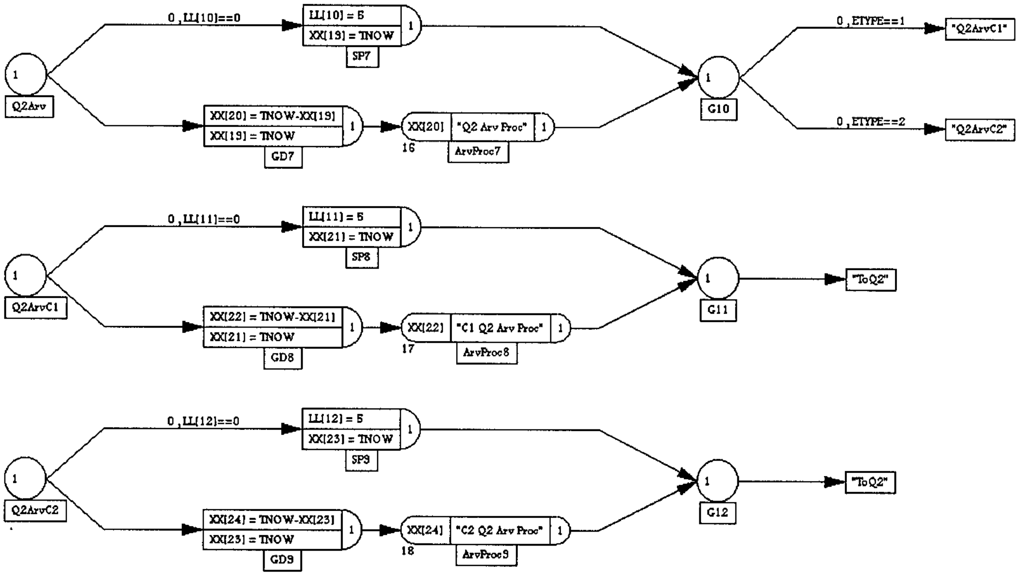

1
ColSerT1
ATRIB[2]
"Q1 Serv T"
1
7
SerQ1
0 , ETYPE==1
ATRIB[2]
"C1 Q1 Serv T"
1
8
SerQ1C1
0 , ETYPE==2
ATRIB[2]
"C2 Q1 Serv T"
1
9
SerQ1C2
1
GoonQ1
"Q1 Wait"

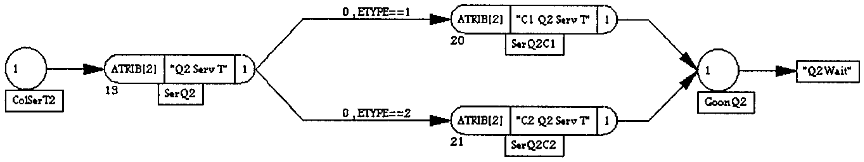

1
Q1 Wait
TNOW-ATRIB[3]-ATRIB[2]
"Q1 Wait T"
1
10
DelayQ1
0 , ETYPE==1
TNOW-ATRIB[3]-ATRIB[2]
"C1 Q1 Wait T"
1
11
C1DelayQ1
"Q1_Cd"
0 , ETYPE==2
TNOW-ATRIB[3]-ATRIB[2]
"C2 Q1 Wait T"
1
12
C2DelayQ1
"Q1_Cd"

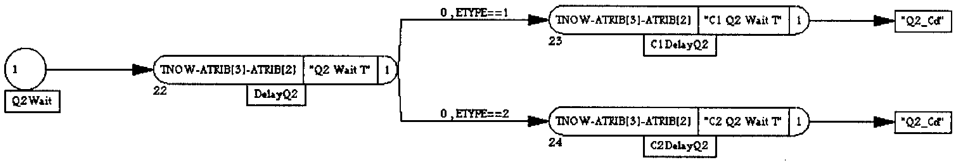

1
Q1_Cd
0 , LL[7]==0
LL[7] = 5
XX[13] = TNOW
1
SetPath1
XX[14] = TNOW-XX[13]
XX[13] = TNOW
1
GetData1
XX[14]
"Q1 Dpt Proc"
1
13
DptProc1
1
G7
0 , ETYPE==1
100
"Q1_Cd1"
0 , ETYPE==2
101
"Q1_Cd2"

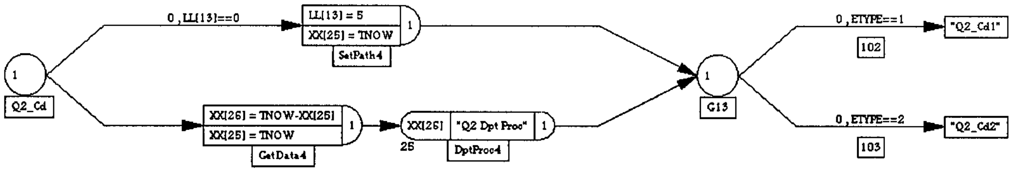

1
Q1_Cd1
0 , LL[8]==0
LL[8] = 5
XX[15] = TNOW
1
SetPath2
XX[16] = TNOW-XX[15]
XX[15] = TNOW
1
GetData2
XX[16]
"C1 Q1 Dpt Proc"
1
14
DptProc2
1
G8
"Choice"

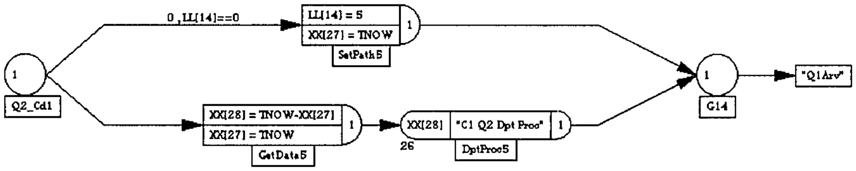

1
Q1_Cd2
0 , LL[9]==0
LL[9] = 5
XX[17] = TNOW
1
SetPath3
XX[18] = TNOW-XX[17]
XX[17] = TNOW
1
GetData3
XX[18]
"C2 Q1 Dpt Proc"
1
15
DptProc3
1
G9
"Choice"

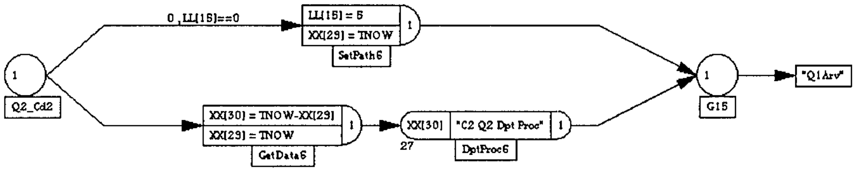

1
Q2Arv
0 ,LL[10]==0
LL[10] = 5
XX[19] = TNOW
1
SF7
XX[20] = TNOW-XX[19]
XX[19] = TNOW
1
GD7
XX[20]
"Q2 Arv Proc"
1
16
ArvProc7
1
G10
0 ,ETYPE==1
"Q2ArvC1"
0 ,ETYPE==2
"Q2ArvC2"
1
Q2ArvC1
0 ,LL[11]==0
LL[11] = 5
XX[21] = TNOW
1
SF8
XX[22] = TNOW-XX[21]
XX[21] = TNOW
1
GD8
XX[22]
"C1 Q2 Arv Proc"
1
17
ArvProc8
1
G11
"ToQ2"
1
Q2ArvC2
0 ,LL[12]==0
LL[12] = 5
XX[23] = TNOW
1
SF9
XX[24] = TNOW-XX[23]
XX[23] = TNOW
1
GD9
XX[24]
"C2 Q2 Arv Proc"
1
18
ArvProc9
1
G12
"ToQ2"

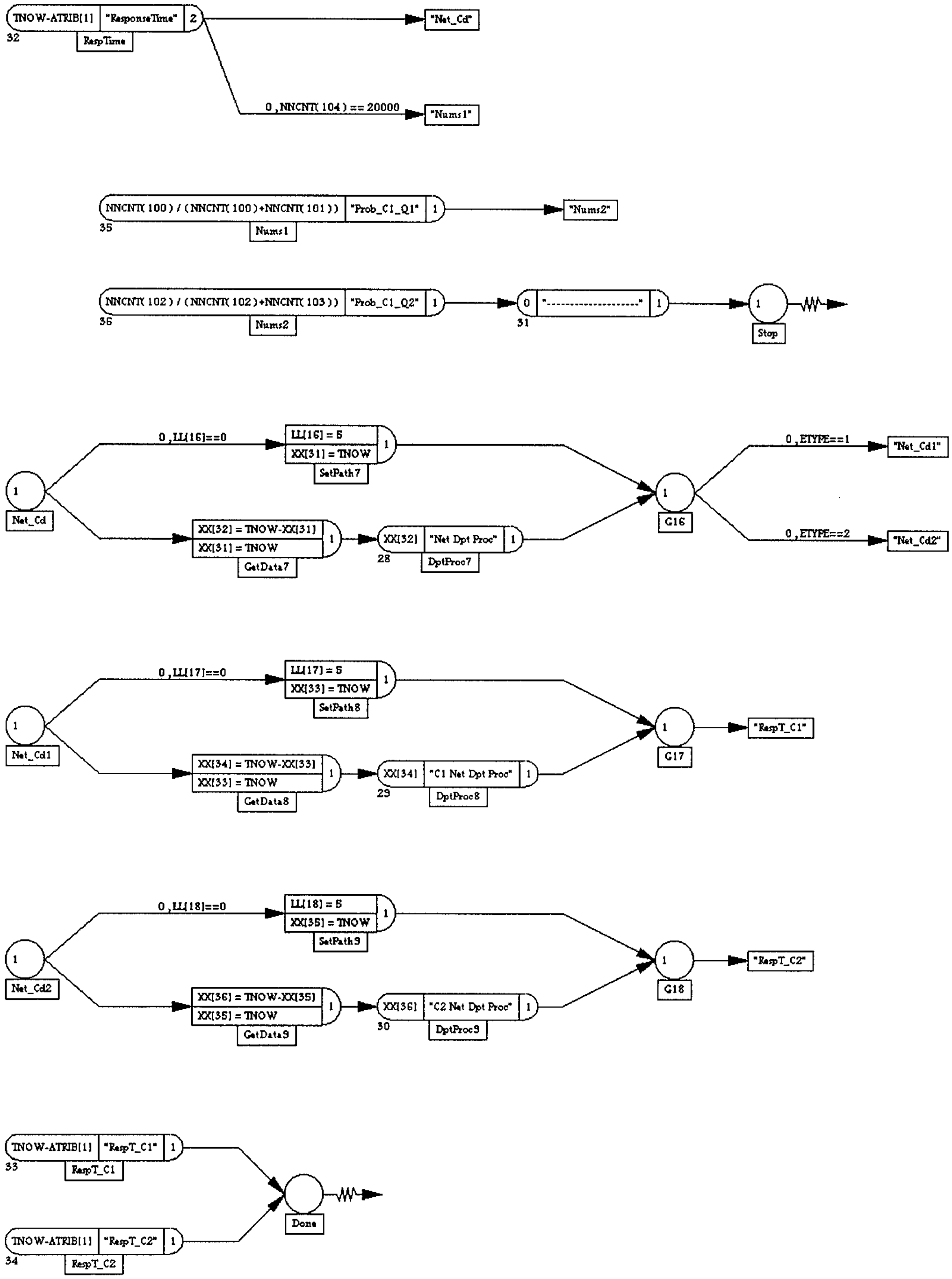

1
ColSerT2
ATRIB[2]
"Q2 Serv T"
1
13
SerQ2
0 , ETYPE==1
ATRIB[2]
"C1 Q2 Serv T"
1
20
SerQ2C1
0 , ETYPE==2
ATRIB[2]
"C2 Q2 Serv T"
1
21
SerQ2C2
1
GoonQ2
"Q2Wait"


1
Q2Wait
TNOW-ATRIB[3]-ATRIB[2]
"Q2 Wait T"
1
22
DelayQ2
0 , ETYPE==1
TNOW-ATRIB[3]-ATRIB[2]
"C1 Q2 Wait T"
1
23
C1DelayQ2
"Q2_Cd"
0 , ETYPE==2
TNOW-ATRIB[3]-ATRIB[2]
"C2 Q2 Wait T"
1
24
C2DelayQ2
"Q2_Cd"


1
Q2_Cd
0 , LL[13]==0
LL[13] = 5
XX[25] = TNOW
1
SetPath4
XX[26] = TNOW-XX[25]
XX[25] = TNOW
1
GetData4
XX[26]
"Q2 Dpt Proc"
1
25
DptProc4
1
G13
0 , ETYPE==1
102
"Q2_Cd1"
0 , ETYPE==2
103
"Q2_Cd2"


1
Q2_Cd1
0 , LL[14]==0
LL[14] = 5
XX[27] = TNOW
1
SetPath5
XX[28] = TNOW-XX[27]
XX[27] = TNOW
1
GetData5
XX[28]
"C1 Q2 Dpt Proc"
1
26
DptProc5
1
G14
"Q1Arv"


1
Q2_Cd2
0 , LL[15]==0
LL[15] = 5
XX[29] = TNOW
1
SetPath6
XX[30] = TNOW-XX[29]
XX[29] = TNOW
1
GetData6
XX[30]
"C2 Q2 Dpt Proc"
1
27
DptProc6
1
G15
"Q1Arv"


TNOW-ATRIB[1]
"ResponseTime"
2
32
RespTime
"Net_Cd"
0 ,NNCNT(104) == 20000
"Nums1"
NNCNT(100) / (NNCNT(100)+NNCNT(101))
"Prob_C1_Q1"
1
35
Nums1
"Nums2"
NNCNT(102) / (NNCNT(102)+NNCNT(103))
"Prob_C1_Q2"
1
36
Nums2
0
"-------------------"
1
31
1
Stop
0 ,LL[16]==0
LL[16] = 5
XX[31] = TNOW
1
SetPath7
1
Net_Cd
XX[32] = TNOW-XX[31]
XX[31] = TNOW
1
GetData7
XX[32]
"Net Dpt Proc"
1
28
DptProc7
1
G16
0 ,ETYPE==1
"Net_Cd1"
0 ,ETYPE==2
"Net_Cd2"
0 ,LL[17]==0
LL[17] = 5
XX[33] = TNOW
1
SetPath8
1
Net_Cd1
XX[34] = TNOW-XX[33]
XX[33] = TNOW
1
GetData8
XX[34]
"C1 Net Dpt Proc"
1
29
DptProc8
1
G17
"RespT_C1"
0 ,LL[18]==0
LL[18] = 5
XX[35] = TNOW
1
SetPath9
1
Net_Cd2
XX[36] = TNOW-XX[35]
XX[35] = TNOW
1
GetData9
XX[36]
"C2 Net Dpt Proc"
1
30
DptProc9
1
G18
"RespT_C2"
TNOW-ATRIB[1]
"RespT_C1"
1
33
RespT_C1
TNOW-ATRIB[1]
"RespT_C2"
1
34
RespT_C2
Done

Appendix C: Two-Node Base Case Simulation Output

Case 1: Base Case Simulation (10,000 Departures with Replication)

```
** AweSim! MULTIPLE RUN SUMMARY REPORT **

  Simulation Project : Thesis, Two Node Test Net
  Modeler : Scott Bellamy
  Date : 06 DEC 98
  Scenario: TWONODE
  Number of runs 25


         ** OBSERVED STATISTICS for scenario TWONODE **


      Label          Mean      Standard    Standard     Minimum     Maximum
                     Value     Deviation    Error       Average     Average
                                                         Value       Value

Net Arv Proc          6.004       0.062       0.012       5.821       6.099
C1 Net Arv Proc       9.972       0.141       0.028       9.553      10.239
C2 Net Arv Proc      15.090       0.226       0.045      14.627      15.495
Q1 Arv Proc           1.807       0.017       0.003       1.770       1.838
C1 Q1 Arv Proc        2.999       0.048       0.010       2.885       3.072
C2 Q1 Arv Proc        4.551       0.092       0.018       4.362       4.727
Q1 Serv T             0.500       0.002       0.000       0.495       0.504
C1 Q1 Serv T          0.500       0.002       0.000       0.494       0.505
C2 Q1 Serv T          0.501       0.005       0.001       0.492       0.511
Q1 Wait T             0.191       0.009       0.002       0.175       0.206
C1 Q1 Wait T          0.191       0.008       0.002       0.176       0.203
C2 Q1 Wait T          0.191       0.011       0.002       0.174       0.213
Q1 Dpt Proc           1.807       0.017       0.003       1.770       1.838
C1 Q1 Dpt Proc        2.999       0.048       0.010       2.885       3.072
C2 Q1 Dpt Proc        4.551       0.092       0.018       4.362       4.727
Q2 Arv Proc           2.586       0.030       0.006       2.521       2.645
C1 Q2 Arv Proc        4.288       0.081       0.016       4.095       4.409
C2 Q2 Arv Proc        6.516       0.168       0.034       6.172       6.807
Q2 Serv T             0.333       0.002       0.000       0.329       0.337
C1 Q2 Serv T          0.333       0.002       0.000       0.329       0.336
C2 Q2 Serv T          0.334       0.004       0.001       0.328       0.340
Q2 Wait T             0.049       0.002       0.000       0.045       0.053
C1 Q2 Wait T          0.049       0.003       0.001       0.045       0.054
C2 Q2 Wait T          0.050       0.003       0.001       0.045       0.056
Q2 Dpt Proc           2.586       0.030       0.006       2.521       2.645
C1 Q2 Dpt Proc        4.288       0.081       0.016       4.095       4.409
C2 Q2 Dpt Proc        6.517       0.168       0.034       6.172       6.807
Net Dpt Proc          6.004       0.062       0.012       5.821       6.098
C1 Net Dpt Proc       9.972       0.141       0.028       9.554      10.239
C2 Net Dpt Proc      15.091       0.227       0.045      14.622      15.497
----------------      0.000       0.000       0.000       0.000       0.000
ResponseTime          3.185       0.059       0.012       3.057       3.289
RespT_C1              3.186       0.063       0.013       3.071       3.285
RespT_C2              3.184       0.088       0.018       3.035       3.343
Prob_C1_Q1            0.603       0.007       0.001       0.592       0.619
Prob_C1_Q2            0.603       0.009       0.002       0.588       0.624
```

```
          ** FILE STATISTICS for scenario TWONODE **

   File      Label or      Average    Standard    Standard     Maximum
  Number   Input Location  Length     Deviation     Error      Average
                                                               Length

       1 QUEUE     Q1         0.106       0.005       0.001       0.115
       2 QUEUE     Q2         0.019       0.001       0.000       0.021

   File     Average
  Number   Wait Time

       1        0.191
       2        0.049

          ** SERVICE ACTIVITY STATISTICS for scenario TWONODE **

 Activity    Label or       Server    Average    Standard    Standard
  Number   Input Location  Capacity Utilization  Deviation     Error

       1 Q1                     1        0.277       0.003       0.001
       2 Q2                     1        0.129       0.001       0.000
```

Case 2: Base Case Simulation (200,000 Departures, One Run)

```
** AweSim SUMMARY REPORT **

  Simulation Project : Thesis, Two Node Test Net
  Modeler : Scott Bellamy
  Date : 06 DEC 98
  Scenario : TWONODE

  Run number 1 of 1
  Current simulation time     : 1199988.269244
  Statistics cleared at time  : 0.000000

         ** OBSERVED STATISTICS REPORT for scenario TWONODE **

     Label            Mean      Standard    Number of     Minimum      Maximum
                      Value     Deviation  Observations    Value        Value

Net Arv Proc           6.000      6.021      199999        0.000       84.567
C1 Net Arv Proc        9.971     10.015      120342        0.000      112.029
C2 Net Arv Proc       15.063     15.024       79656        0.000      164.381
Q1 Arv Proc            1.809      3.520      663326        0.000       78.780
C1 Q1 Arv Proc         3.018      6.297      397663        0.000      104.171
C2 Q1 Arv Proc         4.517      9.803      265662        0.000      153.202
Q1 Serv T              0.501      0.501      663327        0.000        6.569
C1 Q1 Serv T           0.501      0.501      397664        0.000        6.025
C2 Q1 Serv T           0.501      0.501      265663        0.000        6.569
Q1 Wait T              0.192      0.477      663327       -0.000        8.606
C1 Q1 Wait T           0.191      0.475      397664       -0.000        8.606
C2 Q1 Wait T           0.193      0.479      265663       -0.000        7.981
Q1 Dpt Proc            1.809      3.521      663326        0.000       80.164
C1 Q1 Dpt Proc         3.018      6.297      397663        0.000      104.339
C2 Q1 Dpt Proc         4.517      9.805      265662        0.000      152.020
Q2 Arv Proc            2.590      5.290      463326        0.000      104.708
C1 Q2 Arv Proc         4.327      9.320      277320        0.000      168.858
C2 Q2 Arv Proc         6.451     14.418      186005        0.000      228.394
Q2 Serv T              0.333      0.334      463327        0.000        4.265
C1 Q2 Serv T           0.333      0.333      277321        0.000        4.265
C2 Q2 Serv T           0.334      0.335      186006        0.000        3.642
Q2 Wait T              0.050      0.189      463327       -0.000        4.079
C1 Q2 Wait T           0.050      0.189      277321       -0.000        4.079
C2 Q2 Wait T           0.049      0.188      186006       -0.000        3.631
Q2 Dpt Proc            2.590      5.290      463326        0.000      104.821
C1 Q2 Dpt Proc         4.327      9.320      277320        0.000      168.620
C2 Q2 Dpt Proc         6.451     14.419      186005        0.000      228.009
Net Dpt Proc           6.000      6.017      199999        0.000       80.164
C1 Net Dpt Proc        9.971     10.012      120342        0.000      108.220
C2 Net Dpt Proc       15.064     15.015       79656        0.000      157.084
----------------       0.000      0.000           1        0.000        0.000
ResponseTime           3.185      3.469      200000        0.000       52.783
RespT_C1               3.168      3.440      120343        0.000       48.741
RespT_C2               3.210      3.512       79657        0.000       52.783
Prob_C1_Q1             0.599      0.000           1        0.599        0.599
Prob_C1_Q2             0.599      0.000           1        0.599        0.599
```

```
          ** FILE STATISTICS REPORT for scenario TWONODE **

  File        Label or       Average     Standard   Maximum  Current    Average
 Number    Input Location    Length     Deviation   Length   Length   Wait Time

       1 QUEUE     Q1          0.106       0.419        9        0        0.192
       2 QUEUE     Q2          0.019       0.157        6        0        0.050
       0 Event Calendar        3.406       0.559        5        3        3.080

          ** SERVICE ACTIVITY STATISTICS REPORT for scenario TWONODE **

Activity      Label or      Server    Entity      Average    Standard
 Number    Input Location  Capacity    Count    Utilization  Deviation


       1 Q1                      1     663327        0.277        0.447
       2 Q2                      1     463327        0.129        0.335
```

Appendix D: Reconstruction of Queueing Network Analyzer (QNA)

Required Input:
ORIGIN := 1

Number of Queue Nodes: $n := 2$

External Arrival Rate at Node j:

Multiple Streams or Classes Entering Node j:

$$E := \begin{bmatrix} \frac{1}{10} & \frac{1}{15} \\ 0 & 0 \end{bmatrix}$$

Aggregated External Rates for Node j:

$k := 1 .. \mathrm{rows}(E)$

$$\Lambda_{o_k} := \sum_{j=1}^{\mathrm{cols}(E)} E_{k,j} \qquad \Lambda_o = \begin{bmatrix} \lambda_{01} \\ \lambda_{02} \\ \ldots \end{bmatrix}$$

Squared Coefficient of Variation (SCV) of External Arrival Processes, Co:

$$Co := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(Note: Superscript 2 omitted due to Mathcad problematics, interpret as C^2)

(Note: For multiple classes, or non-Poisson arrivals, these need not be one.)

Routing Matrix (Node i to j):

$$Q := \begin{bmatrix} 0 & 0.7 \\ 1 & 0 \end{bmatrix} \qquad Q = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}$$

Service Rate at Node j:

$$\mu := \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

(Note: For Class-dependent service rates, this needs to be the weighted rate.)

Squared Coefficient of Variation (SCV) of Service Processes, Cs:

$$Cs := \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(Note: Superscript 2 omitted due to Mathcad problematics, interpret as C ^{2})

(Note: For Class-dependent service rates, this needs to be a properly weighted value.)

Create Identity Matrix for Later Use:

$I := \text{identity}(n)$

Traffic Rate Equations:

Internal Arrival Rate at Node j:

$$\Lambda := \left[\Lambda_o^{T} \cdot (I - Q)^{-1}\right]^{T}$$

$$\Lambda = \begin{bmatrix} 0.5556 \\ 0.3889 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

Arc Rates (i to j):

$$l := 1 .. \mathrm{cols}(Q)$$

$$\Lambda_{ij_{k,l}} := \Lambda_k \cdot Q_{k,l} \qquad \lambda_{ij} = \lambda_i \cdot q_{ij}$$

$$\Lambda_{ij} = \begin{bmatrix} 0 & 0.3889 \\ 0.3889 & 0 \end{bmatrix}$$

Internal Proportions Arriving at Node j (perspective is to j from i):

$$P_{k,l} := \frac{\Lambda_{ij_{k,l}}}{\Lambda_l} \qquad p_{ij} = \frac{\lambda_{ij}}{\lambda_j}$$

$$P = \begin{bmatrix} 0 & 1 \\ 0.7 & 0 \end{bmatrix}$$

External Proportions Arriving at Node j:

$$Po := \overrightarrow{\left(\frac{\Lambda_o}{\Lambda}\right)}$$

$$Po = \begin{bmatrix} 0.3 \\ 0 \end{bmatrix}$$

Stacked Matrix of Proportions:

$$P := \mathrm{stack}\left(Po^{T}, P\right)$$

Traffic Intensity at Node j:

$$\rho := \overrightarrow{\left(\frac{\Lambda}{\mu}\right)}$$

Traffic Variability Equations:

Weighting Functions:

$j := 1 .. n$

$$\upsilon_j := \left[\sum_{i=1}^{n+1} \left(P_{i,j}\right)^2 \right]^{-1}$$

$$\omega_j := \left[1 + 4 \cdot \left(1 - \rho_j\right)^2 \cdot \left(\upsilon_j - 1\right) \right]^{-1}$$

$$a_j := 1 + \omega_j \cdot \left[\left(Po_j \cdot Co_j - 1\right) + \sum_{i=1}^{n} P_{i,j} \cdot \left[\left(1 - Q_{i,j}\right) + Q_{i,j} \cdot \left(\rho_i\right)^2 \cdot Cs_i \right] \right]$$

$i := 1 .. n$

$$b_{i,j} := \omega_j \cdot P_{i,j} \cdot Q_{i,j} \cdot \left[1 - \left(\rho_i\right)^2 \right]$$

Interim Results:

$$\upsilon = \begin{bmatrix} 1.7241 \\ 1 \end{bmatrix} \qquad \omega = \begin{bmatrix} 0.3983 \\ 1 \end{bmatrix}$$

$$a = \begin{bmatrix} 0.7259 \\ 0.354 \end{bmatrix} \qquad b = \begin{bmatrix} 0 & 0.646 \\ 0.2741 & 0 \end{bmatrix}$$

Estimated Internal Arrival SCV at Node j:

$$Ca := \left[(a)^T \cdot (I - b)^{-1} \right]^T$$

$$Ca = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(Note: Superscript 2 omitted due to Mathcad problematics, interpret as C^2)

Congestion Measures:

Expected Waiting Time at Node j:

$$EW_j := \frac{\rho_j}{\mu_j \cdot \left(1 - \rho_j\right)} \cdot \frac{\left(Ca_j + Cs_j\right)}{2}$$

Expected Number Waiting at Node j:

$$EN := \overrightarrow{(\Lambda \cdot EW)}$$

Expected Number of Visits to Node j:

Total External Arrival Rate:

$$\lambda_o := \sum_{i=1}^{n} \Lambda_{o_i}$$

$$\lambda_o = 0.1667$$

Expected Number of Visits:

$$EV := \frac{1}{\lambda_o} \cdot \Lambda$$

Expected Response Time:

$$ER_j := EV_j \cdot \left(\frac{1}{\mu_j} + EW_j\right)$$

$$ER := \sum_{i=1}^{n} ER_i$$

QNA Summary:

Number of Queue Nodes: $n = 2$

Input External Arrival Rate and SCV at Node j:

$$\Lambda_0 = \begin{bmatrix} 0.1667 \\ 0 \end{bmatrix} \qquad Co = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Routing Matrix (node i to j):

$$Q = \begin{bmatrix} 0 & 0.7 \\ 1 & 0 \end{bmatrix}$$

Internal Arrival Rate and Estimated SCV at Node j:

$$\Lambda = \begin{bmatrix} 0.5556 \\ 0.3889 \end{bmatrix} \qquad Ca = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Input Service Rate and SCV at Node j:

$$\mu = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \qquad Cs = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Traffic Intensity at Node j:

$$\rho = \begin{bmatrix} 0.2778 \\ 0.1296 \end{bmatrix}$$

Expected Number at Node j:

$$EN = \begin{bmatrix} 0.1068 \\ 0.0193 \end{bmatrix}$$

Expected Waiting Time at Node j:

$$EW = \begin{bmatrix} 0.1923 \\ 0.0496 \end{bmatrix}$$

Expected Number of Visits to Node j:

$$EV = \begin{bmatrix} 3.3333 \\ 2.3333 \end{bmatrix}$$

Expected Response Time: $ER = 3.2013$

Appendix E: Two-Node Class-Dependent Simulation Output

Case 1: Different Service Rates at Queue 1 (200,000 Departures)

```
** AweSim SUMMARY REPORT **

  Simulation Project : Thesis, Two Node Test Net
  Modeler : Scott Bellamy
  Date : 06 DEC 98
  Scenario : TWONODE, Case 2

  Run number 1 of 1
  Current simulation time    : 1199988.296267
  Statistics cleared at time : 0.000000


         ** OBSERVED STATISTICS REPORT for scenario TWONODE **

     Label          Mean      Standard    Number of     Minimum      Maximum
                    Value     Deviation  Observations    Value        Value

Net Arv Proc           6.000      6.021      199999        0.000       84.567
C1 Net Arv Proc        9.971     10.015      120342        0.000      112.029
C2 Net Arv Proc       15.063     15.024       79656        0.000      164.381
Q1 Arv Proc            1.809      3.215      663326        0.000       74.043
C1 Q1 Arv Proc         3.016      5.745      397922        0.000      102.488
C2 Q1 Arv Proc         4.521      9.523      265403        0.000      149.279
Q1 Serv T              0.801      0.873      663327        0.000       12.050
C1 Q1 Serv T           1.001      1.002      397923        0.000       12.050
C2 Q1 Serv T           0.501      0.501      265404        0.000        6.569
Q1 Wait T              0.682      1.314      663327       -0.000       17.531
C1 Q1 Wait T           0.719      1.347      397923       -0.000       17.531
C2 Q1 Wait T           0.626      1.261      265404       -0.000       16.690
Q1 Dpt Proc            1.809      3.229      663326        0.000       76.022
C1 Q1 Dpt Proc         3.016      5.733      397922        0.000      102.295
C2 Q1 Dpt Proc         4.521      9.566      265403        0.000      148.098
Q2 Arv Proc            2.590      4.900      463326        0.000       97.329
C1 Q2 Arv Proc         4.323      8.640      277579        0.000      166.798
C2 Q2 Arv Proc         6.460     14.153      185746        0.000      228.394
Q2 Serv T              0.333      0.334      463327        0.000        4.265
C1 Q2 Serv T           0.333      0.333      277580        0.000        4.265
C2 Q2 Serv T           0.334      0.334      185747        0.000        3.642
Q2 Wait T              0.052      0.194      463327       -0.000        4.163
C1 Q2 Wait T           0.045      0.179      277580       -0.000        4.163
C2 Q2 Wait T           0.063      0.213      185747       -0.000        3.486
Q2 Dpt Proc            2.590      4.899      463326        0.000       97.442
C1 Q2 Dpt Proc         4.323      8.638      277579        0.000      166.210
C2 Q2 Dpt Proc         6.460     14.155      185746        0.000      228.009
Net Dpt Proc           6.000      6.057      199999        0.000       79.502
C1 Net Dpt Proc        9.971      9.964      120342        0.000      108.106
C2 Net Dpt Proc       15.064     15.133       79656        0.000      153.161
----------------       0.000      0.000           1        0.000        0.000
ResponseTime           5.811      6.923      200000        0.000      135.953
RespT_C1               6.559      7.491      120343        0.000      135.953
RespT_C2               4.681      5.782       79657        0.000      117.319
Prob_C1_Q1             0.600      0.000           1        0.600        0.600
Prob_C1_Q2             0.599      0.000           1        0.599        0.599
```

```
                ** FILE STATISTICS REPORT for scenario TWONODE **

      File        Label or        Average     Standard   Maximum  Current    Average
     Number    Input Location     Length      Deviation   Length   Length   Wait Time

           1 QUEUE     Q1             0.377       0.941       14        0       0.682
           2 QUEUE     Q2             0.020       0.162        6        0       0.052
           0 Event Calendar           3.571       0.598        5        3       3.230

                ** SERVICE ACTIVITY STATISTICS REPORT for scenario TWONODE **

    Activity      Label or      Server    Entity      Average     Standard
     Number    Input Location  Capacity    Count    Utilization  Deviation


           1 Q1                        1    663327       0.443        0.497
           2 Q2                        1    463327       0.129        0.335
```

Case 1a: Different Service Rates at Queue 1 (10,000 Departures, 10 Replications)

```
        ** AweSim! MULTIPLE RUN SUMMARY REPORT **

  Simulation Project : Thesis, Two Node Test Net
  Modeler : Scott Bellamy
  Date : 06 DEC 98
  Scenario: TWONODE
  Number of runs 10


         ** OBSERVED STATISTICS for scenario TWONODE **

      Label              Mean       Standard    Standard     Minimum     Maximum
                         Value      Deviation    Error       Average     Average
                                                              Value       Value

Net Arv Proc             6.000       0.037       0.012       5.908       6.030
C1 Net Arv Proc          9.972       0.107       0.034       9.735      10.094
C2 Net Arv Proc         15.066       0.170       0.054      14.815      15.408
Q1 Arv Proc              1.809       0.014       0.004       1.786       1.830
C1 Q1 Arv Proc           3.012       0.038       0.012       2.948       3.074
C2 Q1 Arv Proc           4.531       0.066       0.021       4.446       4.637
Q1 Serv T                0.801       0.003       0.001       0.797       0.807
C1 Q1 Serv T             1.000       0.006       0.002       0.988       1.009
C2 Q1 Serv T             0.501       0.002       0.001       0.495       0.505
Q1 Wait T                0.681       0.024       0.007       0.652       0.721
C1 Q1 Wait T             0.718       0.032       0.010       0.680       0.770
C2 Q1 Wait T             0.625       0.012       0.004       0.609       0.648
Q1 Dpt Proc              1.809       0.014       0.004       1.786       1.830
C1 Q1 Dpt Proc           3.012       0.038       0.012       2.948       3.074
C2 Q1 Dpt Proc           4.531       0.066       0.021       4.446       4.637
Q2 Arv Proc              2.590       0.025       0.008       2.560       2.627
C1 Q2 Arv Proc           4.315       0.063       0.020       4.229       4.420
C2 Q2 Arv Proc           6.480       0.113       0.036       6.344       6.681
Q2 Serv T                0.333       0.001       0.000       0.332       0.335
C1 Q2 Serv T             0.333       0.002       0.001       0.331       0.336
C2 Q2 Serv T             0.334       0.002       0.001       0.330       0.337
Q2 Wait T                0.052       0.001       0.000       0.051       0.053
C1 Q2 Wait T             0.045       0.001       0.000       0.043       0.047
C2 Q2 Wait T             0.063       0.001       0.000       0.061       0.066
Q2 Dpt Proc              2.590       0.025       0.008       2.560       2.627
C1 Q2 Dpt Proc           4.315       0.063       0.020       4.229       4.419
C2 Q2 Dpt Proc           6.480       0.113       0.036       6.344       6.681
Net Dpt Proc             6.000       0.037       0.012       5.908       6.030
C1 Net Dpt Proc          9.972       0.106       0.034       9.735      10.094
C2 Net Dpt Proc         15.067       0.172       0.054      14.815      15.413
----------------         0.000       0.000       0.000       0.000       0.000
ResponseTime             5.810       0.100       0.032       5.677       5.982
RespT_C1                 6.564       0.147       0.047       6.372       6.835
RespT_C2                 4.670       0.068       0.021       4.594       4.790
Prob_C1_Q1               0.601       0.005       0.002       0.592       0.608
Prob_C1_Q2               0.600       0.006       0.002       0.591       0.610
```

```
         ** FILE STATISTICS for scenario TWONODE **

  File       Label or       Average    Standard    Standard     Maximum
 Number   Input Location    Length     Deviation    Error       Average
                                                                Length

       1 QUEUE    Q1            0.377       0.013       0.004       0.400
       2 QUEUE    Q2            0.020       0.000       0.000       0.021

  File      Average
 Number    Wait Time


       1       0.681
       2       0.052

         ** SERVICE ACTIVITY STATISTICS for scenario TWONODE **

Activity     Label or      Server    Average    Standard    Standard
 Number   Input Location  Capacity Utilization  Deviation     Error

       1 Q1                      1       0.443       0.004       0.001
       2 Q2                      1       0.129       0.001       0.000
```

Case 2: Different Service Rates at Both Queues (50,000 Departures, 10 Replications)

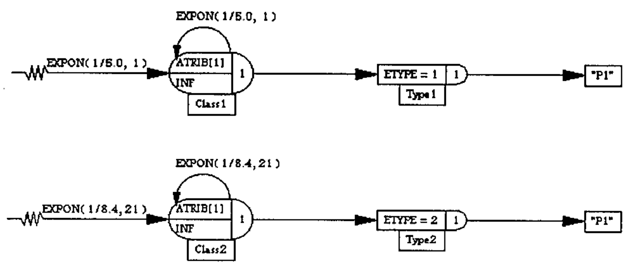
```
** AweSim! MULTIPLE RUN SUMMARY REPORT **

  Simulation Project : Thesis, Two Node Test Net
  Modeler : Scott Bellamy
  Date : 06 DEC 98
  Scenario: TWONODE
  Number of runs 10


          ** OBSERVED STATISTICS for scenario TWONODE **

       Label              Mean      Standard    Standard    Minimum     Maximum
                          Value     Deviation    Error      Average     Average
                                                             Value       Value

Net Arv Proc              5.999       0.018       0.006       5.964       6.021
C1 Net Arv Proc           9.988       0.042       0.013       9.914      10.057
C2 Net Arv Proc          15.024       0.117       0.037      14.841      15.191
Q1 Arv Proc               1.802       0.007       0.002       1.790       1.814
C1 Q1 Arv Proc            2.999       0.020       0.006       2.970       3.032
C2 Q1 Arv Proc            4.516       0.034       0.011       4.442       4.563
Q1 Serv T                 0.801       0.002       0.000       0.798       0.803
C1 Q1 Serv T              1.001       0.003       0.001       0.996       1.005
C2 Q1 Serv T              0.501       0.002       0.001       0.497       0.503
Q1 Wait T                 0.697       0.012       0.004       0.683       0.714
C1 Q1 Wait T              0.759       0.015       0.005       0.738       0.777
C2 Q1 Wait T              0.605       0.013       0.004       0.583       0.624
Q1 Dpt Proc               1.802       0.007       0.002       1.790       1.814
C1 Q1 Dpt Proc            2.999       0.020       0.006       2.971       3.032
C2 Q1 Dpt Proc            4.516       0.034       0.011       4.442       4.563
Q2 Arv Proc               2.576       0.014       0.004       2.553       2.596
C1 Q2 Arv Proc            4.286       0.037       0.012       4.231       4.343
C2 Q2 Arv Proc            6.456       0.053       0.017       6.340       6.521
Q2 Serv T                 0.600       0.003       0.001       0.594       0.605
C1 Q2 Serv T              0.333       0.001       0.000       0.332       0.334
C2 Q2 Serv T              1.002       0.003       0.001       0.996       1.005
Q2 Wait T                 0.225       0.003       0.001       0.221       0.230
C1 Q2 Wait T              0.191       0.003       0.001       0.186       0.196
C2 Q2 Wait T              0.276       0.004       0.001       0.266       0.281
Q2 Dpt Proc               2.576       0.014       0.004       2.553       2.595
C1 Q2 Dpt Proc            4.286       0.037       0.012       4.231       4.343
C2 Q2 Dpt Proc            6.456       0.052       0.017       6.340       6.521
Net Dpt Proc              5.999       0.018       0.006       5.964       6.021
C1 Net Dpt Proc           9.988       0.042       0.013       9.914      10.056
C2 Net Dpt Proc          15.024       0.117       0.037      14.841      15.191
----------------          0.000       0.000       0.000       0.000       0.000
ResponseTime              6.910       0.058       0.018       6.839       7.014
RespT_C1                  7.081       0.081       0.026       6.973       7.217
RespT_C2                  6.652       0.050       0.016       6.573       6.719
Prob_C1_Q1                0.601       0.003       0.001       0.595       0.605
Prob_C1_Q2                0.601       0.003       0.001       0.595       0.606
```

```
         ** FILE STATISTICS for scenario TWONODE **

  File       Label or       Average    Standard    Standard     Maximum
 Number   Input Location    Length     Deviation     Error      Average
                                                                Length

       1 QUEUE    Q1            0.387       0.007       0.002       0.398
       2 QUEUE    Q2            0.087       0.001       0.000       0.089

  File      Average
 Number    Wait Time

       1       0.698
       2       0.225

** SERVICE ACTIVITY STATISTICS for scenario TWONODE **

Activity     Label or      Server    Average    Standard    Standard
 Number   Input Location  Capacity Utilization  Deviation     Error

       1 Q1                      1       0.445       0.002       0.001
       2 Q2                      1       0.233       0.001       0.000
```

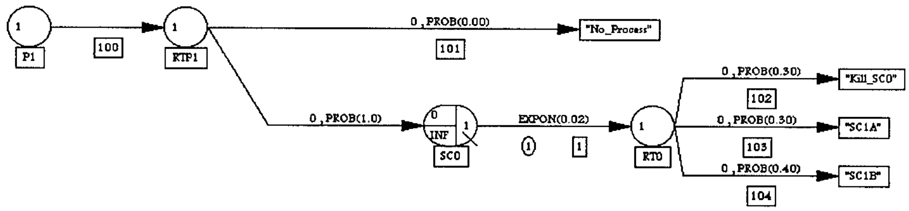
Appendix F: Evaluation Network Simulation Model

Basic Model

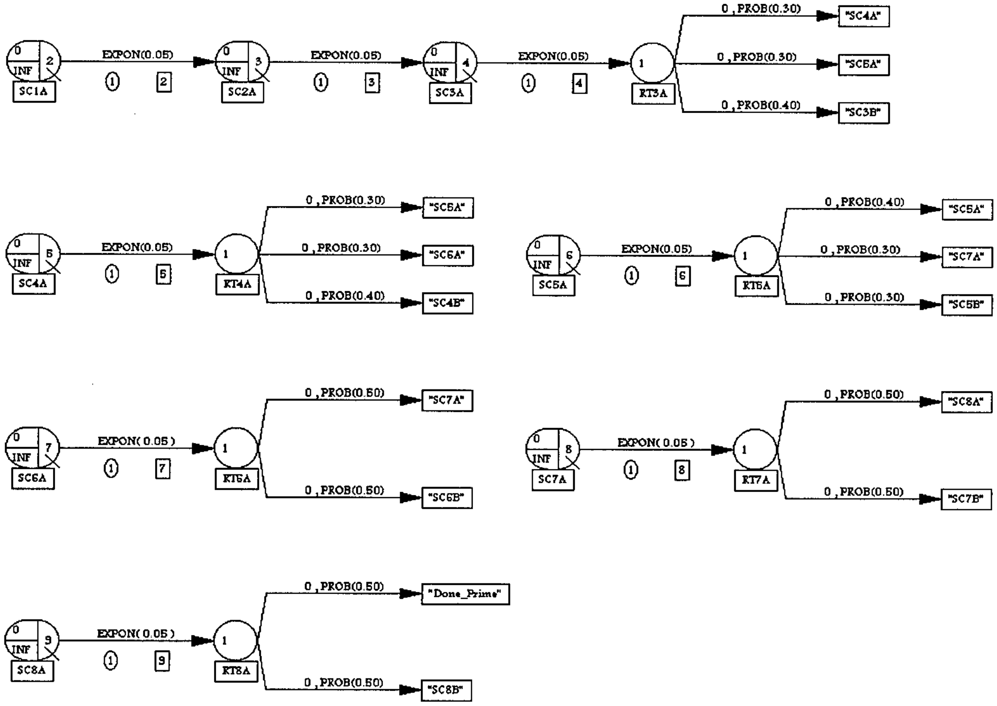
Customer Class Creation



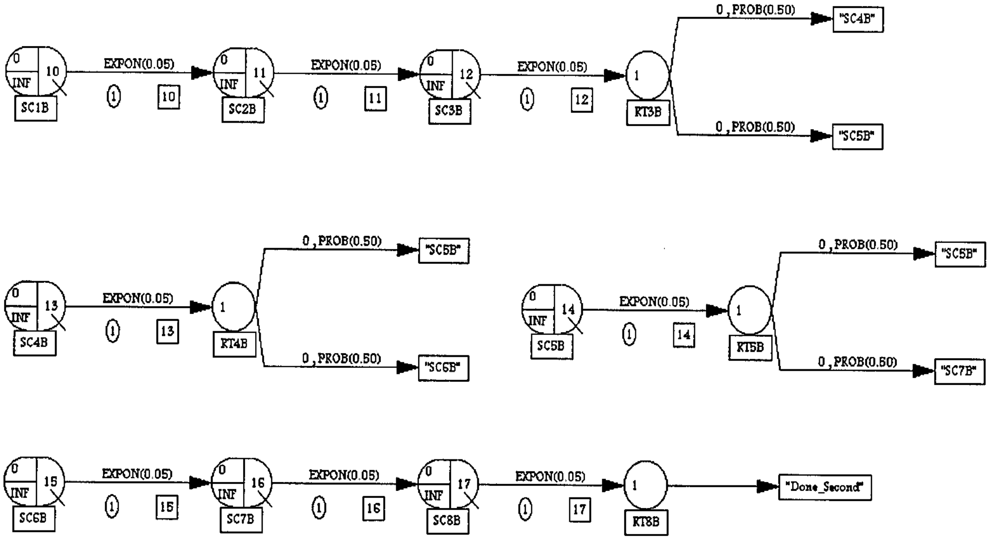
Entry to Network



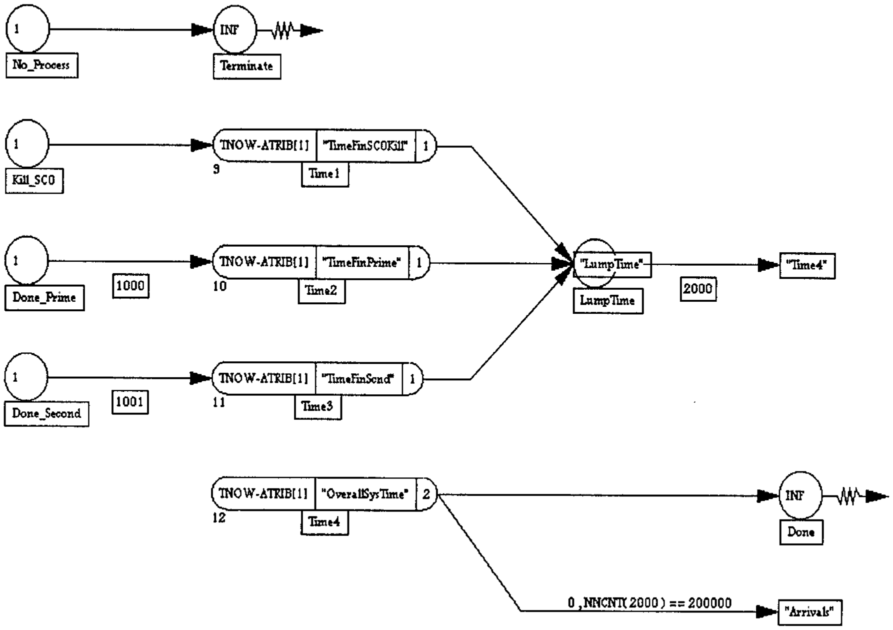
Primary Processing Network



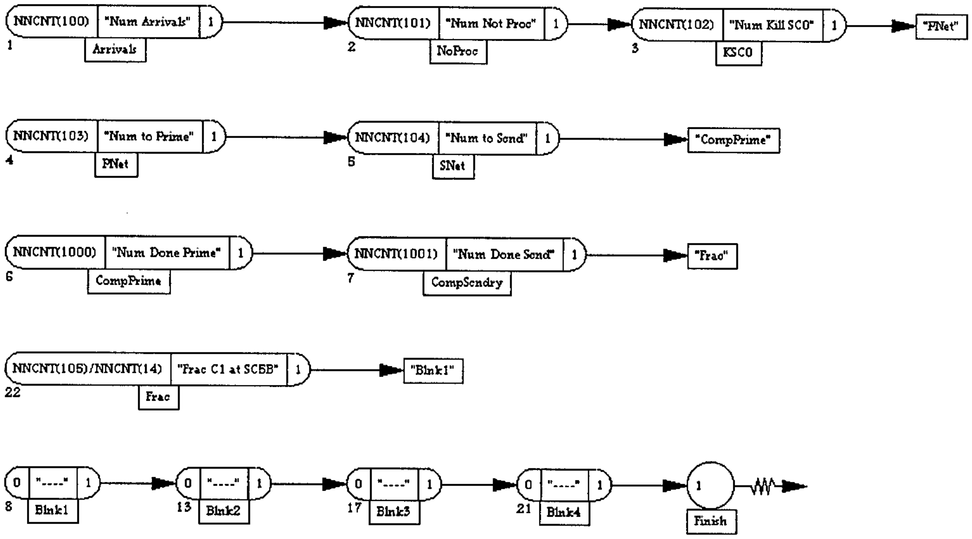
Secondary Processing Network



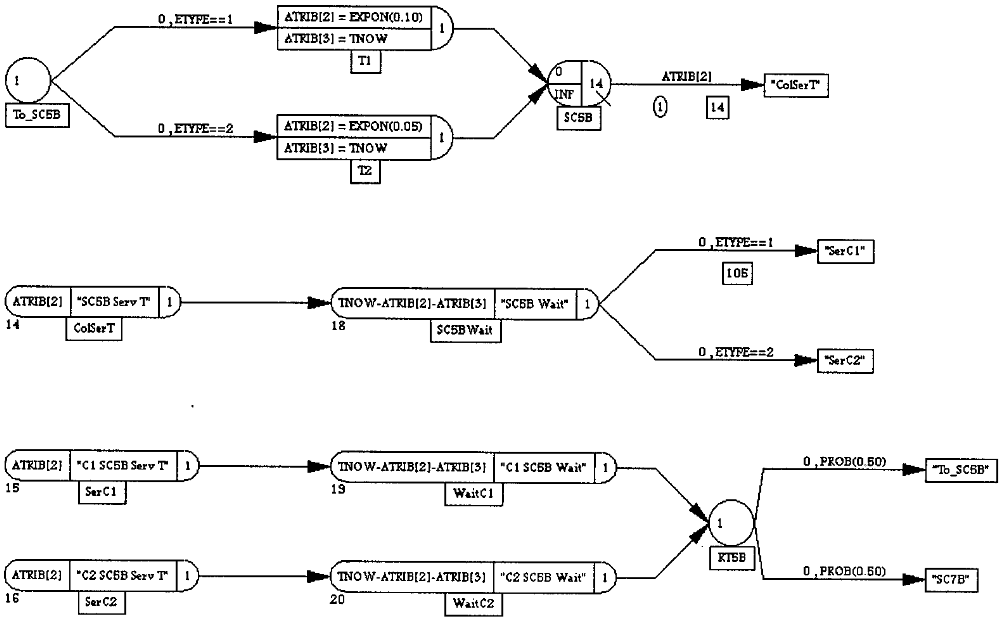
Time Collection



Additional Stats Collection



Modifications



Appendix G: Evaluation Network Simulation Output

Case 1: 200,000 Departures

```
** AweSim SUMMARY REPORT **

  Simulation Project : Thesis, Net 2b
  Modeler : Scott Bellamy
  Date : 23 DEC 98
  Scenario : NET2

  Run number 1 of 1
  Current simulation time    : 14953.345466
  Statistics cleared at time : 0.000000


         ** OBSERVED STATISTICS REPORT for scenario NET2 **

     Label              Mean     Standard   Number of     Minimum     Maximum
                        Value    Deviation  Observations   Value       Value

Num Arrivals      200014.000       0.000          1  200014.000  200014.000
Num Not Proc           0.000       0.000          1       0.000       0.000
Num Kill SC0       60316.000       0.000          1   60316.000   60316.000
Num to Prime       59810.000       0.000          1   59810.000   59810.000
Num to Scnd        79887.000       0.000          1   79887.000   79887.000
Num Done Prime      3630.000       0.000          1    3630.000    3630.000
Num Done Scnd     136054.000       0.000          1  136054.000  136054.000
----                   0.000       0.000          1       0.000       0.000
TimeFinSC0Kill         0.027       0.027      60316       0.000       0.299
TimeFinPrime           0.427       0.169       3630       0.076       1.473
TimeFinScnd            1.237       1.387     136054       0.081      37.008
OverallSysTime         0.858       1.272     200000       0.000      37.008
----                   0.000       0.000          1       0.000       0.000
SC5B Serv T            0.069       0.077     185430       0.000       1.018
C1 SC5B Serv T         0.100       0.101      69615       0.000       1.018
C2 SC5B Serv T         0.050       0.050     115815       0.000       0.580
----                   0.000       0.000          1       0.000       0.000
SC5B Wait              0.476       0.595     185430      -0.000       5.939
C1 SC5B Wait           0.491       0.604      69615       0.000       5.939
C2 SC5B Wait           0.467       0.589     115815      -0.000       5.739
----                   0.000       0.000          1       0.000       0.000
Frac C1 at SC5B        0.375       0.000          1       0.375       0.375


OverallNumInSys       11.473       8.298       0.000      80.000   14953.345
```

```
             ** FILE STATISTICS REPORT for scenario NET2 **

  File         Label or        Average    Standard   Maximum  Current    Average
 Number     Input Location     Length     Deviation   Length   Length   Wait Time

       1 QUEUE     SC0           0.099       0.404       10        0       0.007
       2 QUEUE     SC1A          0.049       0.265        7        0       0.012
       3 QUEUE     SC2A          0.050       0.268        6        0       0.013
       4 QUEUE     SC3A          0.050       0.269        6        0       0.012
       5 QUEUE     SC4A          0.004       0.066        3        0       0.003
       6 QUEUE     SC5A          0.018       0.151        4        0       0.007
       7 QUEUE     SC6A          0.000       0.017        1        0       0.001
       8 QUEUE     SC7A          0.002       0.051        3        0       0.002
       9 QUEUE     SC8A          0.001       0.024        2        0       0.001
      10 QUEUE     SC1B          0.099       0.403        7        0       0.019
      11 QUEUE     SC2B          0.099       0.401        7        0       0.019
      12 QUEUE     SC3B          0.188       0.595        9        0       0.027
      13 QUEUE     SC4B          0.049       0.268        6        0       0.012
      14 QUEUE     SC5B          5.899       7.734       74        8       0.476
      15 QUEUE     SC6B          0.013       0.125        4        0       0.006
      16 QUEUE     SC7B          0.379       0.952       14        0       0.043
      17 QUEUE     SC8B          0.389       0.959       14        0       0.043
       0 Event Calendar          7.085       1.556       15        9       0.072
```

```
             ** SERVICE ACTIVITY STATISTICS REPORT for scenario NET2 **

Activity       Label or       Server     Entity      Average     Standard
 Number     Input Location   Capacity     Count    Utilization  Deviation


       1 SC0                        1     200013       0.268        0.443
       2 SC1A                       1      59809       0.200        0.400
       3 SC2A                       1      59808       0.201        0.400
       4 SC3A                       1      59808       0.200        0.400
       5 SC4A                       1      17590       0.059        0.235
       6 SC5A                       1      38661       0.129        0.335
       7 SC6A                       1       5294       0.018        0.132
       8 SC7A                       1      14371       0.048        0.213
       9 SC8A                       1       7249       0.024        0.153
      10 SC1B                       1      79887       0.268        0.443
      11 SC2B                       1      79887       0.267        0.443
      12 SC3B                       1     104031       0.348        0.476
      13 SC4B                       1      58875       0.197        0.398
      14 SC5B                       1     185430       0.854        0.353
      15 SC6B                       1      32185       0.108        0.310
      16 SC7B                       1     132435       0.444        0.497
      17 SC8B                       1     136054       0.455        0.498
```

Case 2: 250,000 Departures with 10 Replications

```
** AweSim! MULTIPLE RUN SUMMARY REPORT **

  Simulation Project : Thesis, Net 2b
  Modeler : Scott Bellamy
  Date : 23 DEC 98
  Scenario: NET2
  Number of runs 10


         ** OBSERVED STATISTICS for scenario NET2 **


     Label         Mean      Standard    Standard    Minimum     Maximum
                   Value     Deviation    Error      Average     Average
                                                      Value       Value

Num Arrivals     250011.600     8.235       2.604  250003.000  250031.000
Num Not Proc          0.000     0.000       0.000       0.000       0.000
Num Kill SC0      75000.800   245.572      77.657   74518.000   75360.000
Num to Prime      75018.000   260.383      82.340   74660.000   75303.000
Num to Scnd       99992.400   198.329      62.717   99698.000  100278.000
Num Done Prime     4515.700    40.376      12.768    4445.000    4564.000
Num Done Scnd    170483.500   256.540      81.125  170082.000  170962.000
----                  0.000     0.000       0.000       0.000       0.000
TimeFinSC0Kill        0.027     0.000       0.000       0.027       0.028
TimeFinPrime          0.426     0.003       0.001       0.422       0.432
TimeFinScnd           1.226     0.029       0.009       1.188       1.281
OverallSysTime        0.852     0.020       0.006       0.826       0.890
----                  0.000     0.000       0.000       0.000       0.000
SC5B Serv T           0.069     0.000       0.000       0.068       0.069
C1 SC5B Serv T        0.100     0.000       0.000       0.099       0.100
C2 SC5B Serv T        0.050     0.000       0.000       0.050       0.050
----                  0.000     0.000       0.000       0.000       0.000
SC5B Wait             0.465     0.020       0.006       0.439       0.502
C1 SC5B Wait          0.477     0.020       0.006       0.455       0.518
C2 SC5B Wait          0.457     0.020       0.006       0.429       0.492
----                  0.000     0.000       0.000       0.000       0.000
Frac C1 at SC5B       0.373     0.002       0.001       0.370       0.376


OverallNumInSys      11.409     0.263       0.083      11.022      11.935
```

```
            ** FILE STATISTICS for scenario NET2 **

   File        Label or        Average     Standard    Standard     Maximum
  Number    Input Location     Length      Deviation     Error      Average
                                                                    Length

        1 QUEUE      SC0          0.099       0.001       0.000       0.100
        2 QUEUE      SC1A         0.051       0.001       0.000       0.054
        3 QUEUE      SC2A         0.051       0.001       0.000       0.052
        4 QUEUE      SC3A         0.051       0.001       0.000       0.052
        5 QUEUE      SC4A         0.004       0.000       0.000       0.004
        6 QUEUE      SC5A         0.020       0.001       0.000       0.020
        7 QUEUE      SC6A         0.000       0.000       0.000       0.000
        8 QUEUE      SC7A         0.002       0.000       0.000       0.003
        9 QUEUE      SC8A         0.001       0.000       0.000       0.001
       10 QUEUE      SC1B         0.098       0.002       0.001       0.101
       11 QUEUE      SC2B         0.098       0.001       0.000       0.099
       12 QUEUE      SC3B         0.187       0.002       0.001       0.191
       13 QUEUE      SC4B         0.049       0.001       0.000       0.051
       14 QUEUE      SC5B         5.808       0.249       0.079       6.311
       15 QUEUE      SC6B         0.013       0.000       0.000       0.014
       16 QUEUE      SC7B         0.375       0.008       0.003       0.386
       17 QUEUE      SC8B         0.400       0.006       0.002       0.408

   File      Average
  Number    Wait Time


        1       0.007
        2       0.013
        3       0.013
        4       0.013
        5       0.003
        6       0.007
        7       0.001
        8       0.003
        9       0.001
       10       0.018
       11       0.018
       12       0.027
       13       0.012
       14       0.465
       15       0.006
       16       0.042
       17       0.044
```

```
        ** SERVICE ACTIVITY STATISTICS for scenario NET2 **

Activity     Label or      Server    Average    Standard    Standard
 Number   Input Location  Capacity Utilization  Deviation     Error

      1 SC0                    1        0.268       0.001       0.000
      2 SC1A                   1        0.201       0.001       0.000
      3 SC2A                   1        0.201       0.001       0.000
      4 SC3A                   1        0.201       0.001       0.000
      5 SC4A                   1        0.060       0.001       0.000
      6 SC5A                   1        0.130       0.001       0.000
      7 SC6A                   1        0.018       0.000       0.000
      8 SC7A                   1        0.048       0.000       0.000
      9 SC8A                   1        0.024       0.000       0.000
     10 SC1B                   1        0.268       0.001       0.000
     11 SC2B                   1        0.268       0.001       0.000
     12 SC3B                   1        0.348       0.001       0.000
     13 SC4B                   1        0.198       0.001       0.000
     14 SC5B                   1        0.858       0.004       0.001
     15 SC6B                   1        0.108       0.001       0.000
     16 SC7B                   1        0.445       0.002       0.001
     17 SC8B                   1        0.457       0.002       0.001
```

Database File (partial contents):

RUN	ID	MEAN	STDDEV
1	C1 SC5B Wait	0.489111	0.591964
2	C1 SC5B Wait	0.467019	0.527733
3	C1 SC5B Wait	0.455314	0.546207
4	C1 SC5B Wait	0.485863	0.541861
5	C1 SC5B Wait	0.489467	0.564231
6	C1 SC5B Wait	0.486997	0.555241
7	C1 SC5B Wait	0.470935	0.506844
8	C1 SC5B Wait	0.518326	0.610003
9	C1 SC5B Wait	0.454573	0.532559
10	C1 SC5B Wait	0.455114	0.506825
1	C2 SC5B Wait	0.467188	0.579109
2	C2 SC5B Wait	0.448599	0.527040
3	C2 SC5B Wait	0.429397	0.522720
4	C2 SC5B Wait	0.468458	0.546739
5	C2 SC5B Wait	0.474599	0.569887
6	C2 SC5B Wait	0.465969	0.550299
7	C2 SC5B Wait	0.451025	0.504098
8	C2 SC5B Wait	0.492360	0.601518
9	C2 SC5B Wait	0.437241	0.527563
10	C2 SC5B Wait	0.434369	0.501537

Bibliography

1. Albin, S. "Approximating a Point Process by a Renewal Process, II: Superposition Arrival Processes to Queues," Operations Research, Volume 32, No 5, 1984: 1133-1162.

2. Banks, J., J. S. Carson, and B. L. Nelson. Discrete-Event System Simulation (Second Edition). United States: Prentice-Hall, 1996.

3. Baskett, F., K. M. Chandy, R.R. Muntz, and F. Palacios. "Open, Closed and Mixed Networks of queues with Different Classes of Customers," Journal of the Association for Computing Machinery, Volume 22, No 2, April 1975: 248-260.

4. Bronshtein, O. and I. B. Gertsbakh. "An Open Exponential Queuing Network with Limited Waiting Spaces and Losses: A Method of Approximate Analysis," Performance Evaluation, Volume 4, 1984: 31-43.

5. Browne S. and W. Whitt. "Piecewise-Linear Diffusion Processes," in Advance in Queueing: Theory, Methods, and Open Problems. United States: CRC Press, 1995.

6. Burke, P. J. "The Output of a Queueing System," Operations Research, Volume 4, 1956: 699-704.

7. Burke, P. J. "Output Processes in Tandem Queues," in Proceedings of the Symposium on Computer-Communications Networks and Teletraffic. New York: Wiley & Sons, 1972.

8. Chandy, K. M., J. H. Howard, and D. F. Towsley. "Product Form and Local Balance in Queueing Networks," Journal of the Association for Computing Machinery, Volume 24, No 2, April 1977: 250-263.

9. Chandy, K. M. and C. H. Sauer. "Approximate Methods for Analyzing Queueing Network Models of Computing System," Computing Surveys, Volume 10, No 3, 1978: 281-317.

10. Chandy, K. M. and A. J. Martin. "A Characterization of Product-Form Queueing Networks," Journal of the Association for Computing Machinery, Volume 30, No 2, April 1983: 286-299.

11. Conway, A. E. and N. D. Georganas. Queueing Networks—Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation. Cambridge: The MIT Press, 1989.

12. Denning, P. J. and J. P. Buzzen. "The Operational Analysis of Queueing Network Models," Computing Surveys, Volume 10, No 3, September 1978: 225-261.

13. Disney, R. L., D. C. McNickle, and B. Simon. "The M/G/1 Queue with Instantaneous Feedback," Naval Research and Logistics Quarterly, Volume 27, 1980: 635-644.

14. Disney, R. L. and P. C. Kiessler. Traffic Processes in Queueing Networks : A Markov Renewal Approach. Baltimore: Johns Hopkins University Press, 1987.

15. Gelenbe, E. and I. Mitrani. Analysis and Synthesis of Computer Systems. United Kingdom, Academic Press, 1980.

16. Glynn, P. W. "Diffusion Approximations," in Handbook in Operations Research and Management Science: Vol 2. The Netherlands: Elsevier, 1990.

17. Gross, D. and C. M. Harris. Fundamentals of Queueing Theory (Second Edition). United States: Wiley & Sons, 1985.

18. Harrison, J. M. and V. Nguyen. "The QNET Method for Two-Moment Analysis of Open Queueing Networks," Queueing Systems, Volume 6, 1990: 1-32.

19. Harrison, J. M. and V. Nguyen. "Brownian Models of Multiclass Queueing Networks: Current Status and Open Problems," Queueing Systems, Volume 13, 1993: 5-40.

20. Jackson, J. R. "Networks of Waiting Lines," Operations Research, Volume 5, 1957: 518-521.

21. Jackson, J. R. "Job Shop Like Queueing Systems," Management Sciences, Volume 10, No 1, 1963: 131-142.

22. Kelly, F. P. "Networks of Queues with Customers of Different Types," Journal of Applied Probability, Volume 12, 1975: 542-554.

23. Kelly, F. P. and C. N. Laws. "Dynamic Routing in Open Queueing Networks: Brownian Models, Cut Constraints, and Resource Pooling," Queueing Systems, Volume 13, 1993: 47-86.

24. Kobayashi, H. Modeling and Analysis: An Introduction to System Performance Evaluation Methodology. United States: Addison-Wesley, 1978.

25. Kühn, P. J. "Analysis of Complex Queuing Networks by Decomposition," Proceedings of the 8th International Teletraffic Congress, Volume 1, November 1976: 236-1 – 236-8.

26. (Kühn) Kuehn, P. J. "Approximate Analysis of General Queuing Networks by Decomposition," IEEE Transactions on Communications, Volume 27, No 1, January 1979: 113-126.

27. Kulkarni, V. G. Modeling and Analysis of Stochastic Systems. United Kingdom: Chapman and Hall, 1995.

28. Lavenberg, S. S. Computer Performance Modeling Handbook. United States: Academic Press, 1983.

29. Law, A. M., and W. D. Kelton. Simulation Modeling & Analysis (Second Edition). United States: McGraw-Hill, 1991.

30. Lazowska, D. E., J. Zahorhan, G. S. Graham, and K. C. Sevcik. Quantitative System Performance: Computer System Analysis Using Queueing Network Models. New Jersey: Prentice-Hall, 1984.

31. Marshall, K. T. "Some Inequalities in Queueing," Operations Research, Volume 16, 1968: 651-665.

32. Medhi, J. Stochastic Models in Queueing Theory. United States: Academic Press, 1991.

33. Melamed, B. "Sojourn Times in Queueing Networks," Mathematics of Operations Research, Volume 7, 1982: 223-244.

34. Noetzel, A. S. "A Generalized Queueing Discipline for Product Form Networks," Journal of the Association for Computing Machinery, Volume 26, No 4, October 1979: 779-793.

35. Perros, H. G. "A Bibliography of Papers on Queueing Networks with Finite Capacity Queues," Performance Evaluation, Volume 10, 1989: 255-260.

36. Perros, H. G. and T. Altiok (editors). Queueing Networks with Blocking. The Netherlands: Elsevier, 1989.

37. Pritsker, A. A. B., J. J. O'Reilly, and D. K. LaVal. Simulation with Visual SLAM and Awesim. United States: Wiley & Sons, 1997.

38. Reiser, M. and H. Kobayashi. "Accuracy of the Diffusion Approximation for Some Queueing Systems," IBM Journal of Research and Development, Volume 18, 1974: 110-124.

39. Ross, S. M. Introduction to Probability Models (Sixth Edition). UNITED STATES: Academic Press, 1997.

40. Sauer, C. H. and K. M. Chandy. Computer Systems Performance Modeling. UNITED STATES: Prentice-Hall, 1981.

41. Schwartz, M. Telecommunications Networks: Protocols, Modeling, and Analysis. United States: Addison-Wesley, 1987.

42. Seery, A. T. User's Manual for QNA 1.0 . US: AT&T Bell Laboratories, 1984.

43. Sevcik, K. C., A. I. Levy, S. K. Tripathi, and J. L. Zahorjan. "Improving Approximations of Aggregated Queueing Network Subsystems," in Computer Performance. Amsterdam: North Holland, 1977.

44. Shanthikumar, J. and J. A. Buzacott. "Open Queueing Network Models of Dynamic Job Shops," Internal Journal of Production Research, Volume 10, No 3, 1981: 255-266.

45. Shi, L. "Approximate Analysis for Queueing Networks with Finite Capacity and Customer Loss," European Journal of Operations Research, Volume 85, 1995: 178-191.

46. Trivedi, K. Probability and Statistic with Reliability, Queueing, and Computer Science Applications. New Jersey: Prentice-Hall, 1982.

47. Walrand, J. "Queueing Networks," in Handbook in Operations Research and Management Science: Vol 2. The Netherlands: Elsevier, 1990.

48. Whitt, W. "Approximating a Point Process by a Renewal Process, I: Two Basic Methods," Operations Research, Volume 30, No 1, 1982: 125-147.

49. Whitt, W. "The Queueing Network Analyzer," The Bell Systems Technical Journal, Volume 62, No 9, 1983, 2779-2813.

50. Whitt, W. "Performance of the Queueing Network Analyzer," The Bell Systems Technical Journal, Volume 62, No 9, 1983, 2817-2843.

51. Whitt, W. "Towards Better Multi-Class Parametric-Decomposition Approximations for Open Queueing Networks," Annals of Operations Research, Volume 48, 1994: 221-248.

52. Whitt, W. (AT&T), M. Segal (AT&T), and W. D. Compton (Purdue University). Electronic Correspondence. November-December 1998.

VITA

Capt Kelly Scott Bellamy entered Air Force active duty in March 1984. Through the Airman's Education and Commissioning Program, he completed his Bachelor of Science degree in Mechanical Engineering, with a partial minor in Aerospace Engineering, in May 1992. He received his commission in the Air Force in September 1992 from the Officer Training School.

Capt Bellamy's first assignment following commissioning was Weapon Systems Engineer for all airborne platforms in the Space and Special Systems Management Directorate at Warner-Robins Air Logistic Center. Following that, he served as a Basic Research Scientist and Rocket Engine Test Engineer in the Propulsion Directorate at the former Phillips Laboratory (now Air Force Research Laboratory) operating location at Edwards Air Force Base.

He entered the School of Engineering, Air Force Institute of Technology, in August 1997. His assignment upon completion of a Master of Science in Operations Research will be to the Analysis Section of the Air Force Personnel Center.

Permanent Address: 514 High Rock Rd
Stanton KY 40380

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
March 1999

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE

ANALYSIS OF A NON-TRIVIAL QUEUEING NETWORK

5. FUNDING NUMBERS

6. AUTHOR(S)

Kelly Scott Bellamy, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
2950 P Street
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOA/ENS/99M-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Jose C. Belano, Captain, USAF
National Security Agency
9800 Savage Road, Suite 6678
Fort Meade MD 20755-6678

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release: Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In studying complex queueing networks, one generally seeks to employ exact analytic solutions to reduce burden on computational resources. Barring the existence of an exact solution, the alternatives include approximation techniques and simulation. Approximation is the more attractive alternative from a time and effort perspective; however, cases exist that are not amiable to this technique.

This work increases the flexibility of approximation techniques in obtaining estimates of congestion measures for complex, open queueing networks having several customer classes and class-dependent structures. We accomplish this by providing the procedure to aggregate multiple classes into a single class in order to apply an existing approximation technique. The resulting method is shown to yield good agreement with results obtained by simulation.

The immediate application of this work is as a tool to focus a simulation study of multi-class queueing networks. For large networks, reasonable performance estimates can be obtained quickly. Once the basic input parameters are determined, different scenarios may be rapidly evaluated in a fraction of the time needed to modify a typical simulation model. This allows one to check ideas and determine where to invest time and funding when constructing a simulation model to obtain performance estimates on a by-class basis.

14. SUBJECT TERMS

Queueing, Network, Simulation, Approximation

15. NUMBER OF PAGES
137

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL